

Dr. Dobb's Journal of

111 JANUARY 1986
\$2.95 (3.95 CANADA)

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Launching Our Second Decade

Bringing up a
68K Machine
from Scratch

PL/68K:
Is it C or
is it Assembler?

An 8080 Simulator
for the 68K

DOS Shell Notes
on 80286 Big DOS

**10th
Anniversary
Issue**

A S S E M B L Y
N G U A G E



Breakthrough for C Programmers



H.E.L.P. Eliminates Every Bug known to Compilers ... As well as a few other species

H.E.L.P. is a completely interactive C programming environment with three innovative full-sized features that will revolutionize the way you write code.

A Clean Compile — Guaranteed!

Say Good-bye to all compiler-type errors. **H.E.L.P.**'s built-in program checker (which would embarrass LINT) not only hunts down bugs ... explains the proper syntax ... gives examples of usage ... but will even offer suggested corrections. If you want, **H.E.L.P.** will even make the corrections for you ... at the touch of a key.

H.E.L.P. also finds semantic errors as well as poor style and inefficiencies. You can even check the portability of your code!

Multi-Window Editing

Open as many windows as you want ... there's no limitation ... not even your own memory. Because **H.E.L.P.** uses a virtual memory system you can create programs larger than your machine capacity.

H.E.L.P.'s very powerful editor allows you the flexibility to work in several windows ... with several files at the same time.

Circle no. 165 on reader service card

Save Keystrokes

Hundreds of commands are bound to the keyboard to give you fast execution.

Always be in Control

Not only can you develop code in many windows at the same time, but you can show (and refer to) important definitions in one window while creating in another. Or open a window and keep notes about your program ... or type a memo ... or a letter.

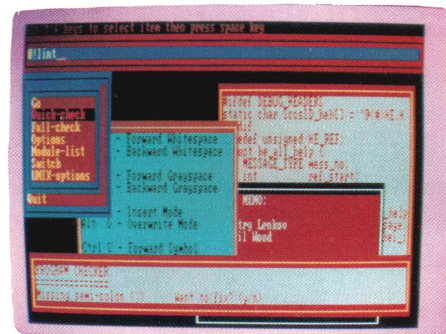
Increase your Productivity by 300% or More

If you are a novice programmer, you'll begin writing code like an advanced programmer much faster with **H.E.L.P.**

Just imagine what **H.E.L.P.** will do for the **ADVANCED PROGRAMMER**.

You'll have more time to become creative with your algorithm (since **H.E.L.P.** will make sure your code compiles the "first time at bat").

H.E.L.P. tracks every step you make. If you are not sure about a command, just press a key, and you'll get the kind of help you need.



Check These Features

- Multi-window environment
- Interactive program checking
- Check syntax, semantic, type usage, intermodule inconsistencies and portability
- Multi-file editing
- Intelligent help subsystem
- User-definable keyboard bindings
- Supports color and monochrome
- **H.E.L.P.** supports the full C Language

NOW IN MS-DOS

EVEREST

Order now \$395

SOLUTIONS

Everest Solutions, Inc.
3350 Scott Boulevard
Building 58
Santa Clara, CA 95051
(408) 986-8977



Optotech, Inc.

The 5¼ inch Optical Disk Drive Is Here!

Optical Disk Drive 5984

- 200 megabytes on a removable cartridge.
- Fast access read and write.
- For PCs and minis.
- Extensive interface software.
- Available Now.

The Preferred Solution For:

- Database—large, portable, indelible and updatable
- Online Mass Storage—integral backup
- Imaging—capacity with removability



Optotech, Inc.

770 Wooten Road
Colorado Springs, CO 80915
303.570.7500 Telex 592966



Good news for software developers:

No royalties on Btrieve®.

Effective January 1, SoftCraft will no longer charge royalties for our Btrieve file management system.

By dropping royalties, we're giving software authors and application developers something to bank on: increased profitability on every copy of a Btrieve application that is used or sold. What you do with Btrieve is up to you. Whether you're developing applications for a handful of users or hundreds, you pay for Btrieve only once—no strings attached. The price remains the same.

With no royalties, doing business with SoftCraft is easier and more profitable than ever. We're showing our appreciation to current Btrieve users, while furthering the rapid expansion of Btrieve applications. We believe this move will reinforce Btrieve's position as the file management standard for IBM PC or AT software developers.

SoftCraft is committed to providing our customers with the comprehensive development tools they need. In the last 18 months alone we've introduced database query and report writing modules for Btrieve applications, as well as local area network and XENIX versions of Btrieve. Our Btrieve environment continues to grow, keeping you in the forefront as future trends emerge.

We're convinced our new "no more royalties" plan will make you—and Btrieve—even more successful.

And that's a great way to start the new business year.



SoftCraft Inc.

P.O. Box 9802 #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

*Suggested retail prices: Btrieve, \$245; Btrieve/N (network version), \$595;
Xtrieve, \$195; Xtrieve/N, \$395; Rtrieve, \$85; Rtrieve/N, \$175.*

Requires PC-DOS, MS-DOS 1.X-3.X or XENIX.

*Btrieve is a registered trademark and Xtrieve and Rtrieve
are trademarks of SoftCraft Inc.*

Circle **no. 113** on reader service card.

Dr. Dobb's Journal of**Software Tools**

FOR THE PROFESSIONAL PROGRAMMER

ARTICLES

COMPILERS: PL/68K by Edward K. Ream **26**
The author of the RED editor could find no acceptable compiler for the 68K, so he wrote one. But PL/68K is not like other compilers.

SYSTEMS: A Simple OS for Real Time Applications by Nick Turner **44**
In setting up a multiuser system, the author found ways to shave many machine cycles off every instruction. Here he explains how he did it and offers some illustrative code.

HARDWARE: Bringing up the 68000 by Alan K. Wilcox **60**
Here's how to get a 68000 running very quickly.

PORTABILITY: COM: An 8080 Simulator for the MC68000 by Jim Cathey **76**
Besides being a useful tool, this program contains some insights gained in the process of developing a 68K application. The author also shows how to extend the simulator for Z80 instructions.

COLUMNS

C CHEST: A Unix-like Shell for MS DOS **18**
by Allen Holub
Allen describes how the shell works on a high level and includes examples of some often-used functions.

16-BIT TOOLBOX: Trojan Horse Programs **118**
by Ray Duncan
These mysterious and destructive programs are appearing on bulletin boards. What can be done to combat them?

FORUM

EDITORIAL: The New Look by Michael Swaine **6**
LETTERS: Comment **8**
by our readers
CARTOON: A Word on Formats by Rand Renfroe **8**
VIEWPOINT: Inefficient C **16**
by Hal Hardenberg
DDJ ON LINE: The Electronic Edition **17**
by Frank DeRose

PROGRAMMERS' SERVICES

PROFESSIONAL PROGRAMMER: A little library of books on the profession **124**
OF INTEREST: New products of interest to programmers **126**
ADVERTISER **128**
INDEX: Where to find that ad

*Is it C or is it assembly?
WordStar on an Atari ST? ►*

Mysterious and dangerous programs ►

What C programmers don't know can hurt them. ►

DDJ is now on CompuServe.

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Launching
Our Second
Decade

Bringing up a
68K Machine
from Scratch

PL/68K:
Is it C or
is it Assembler?

An 8080 Simulator
for the 68K
DOS Shell Notes
on 80286 Big DOS



Our anniversary cover was designed by Shelley Rae Doeden, who also is responsible for the new look of the magazine. Shelley is DDJ's Art Director.

This Issue:

The Motorola 68000 chip is, some say, a programmer's processor. The instruction set is rich and logically constructed, and memory access is simple and capacious. What it lacks, others say, are a standard operating system and development tools. This month we focus on programming tools for the 68K.

Next Issue:

Next month we'll look at structured programming and at some languages that have a reputation for encouraging structured design. We'll publish programming tools in Pascal, Modula-2, and Ada. We'll tell you where to get public-domain Ada utilities to speed Ada software development, and we'll look at a program that ports between dialects of Pascal.

PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

TURBO PASCAL

Turbo ASYNCH™

With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include...

- ◆ Initialization of the COM ports allowing you to set all transmission options.
- ◆ Interrupt processing.
- ◆ Data transfer between circular queues and communications ports.
- ◆ Simultaneous buffered input and output to both COM ports.
- ◆ Transmission speeds up to 9600 Baud.
- ◆ Input and output queues as large as you wish.
- ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for \$99.95 each, and they may be ordered directly from Blaise Computing Inc. To order, call (415) 540-5441.

◆
BLAISE COMPUTING INC.

BLAISE

NEW!

Turbo POWER TOOLS™

Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including...

- ◆ Extensive string handling to complement the powerful Turbo Pascal functions.
- ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls.
- ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT.
- ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application.
- ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is _____ for
☐ Turbo ASYNCH ☐ Turbo POWER TOOLS. (CA residents add
6½% Sales Tax. All orders add \$6.00 for shipping.)

Name: _____ Phone: _____

Shipping Address: _____ State: _____ Zip: _____

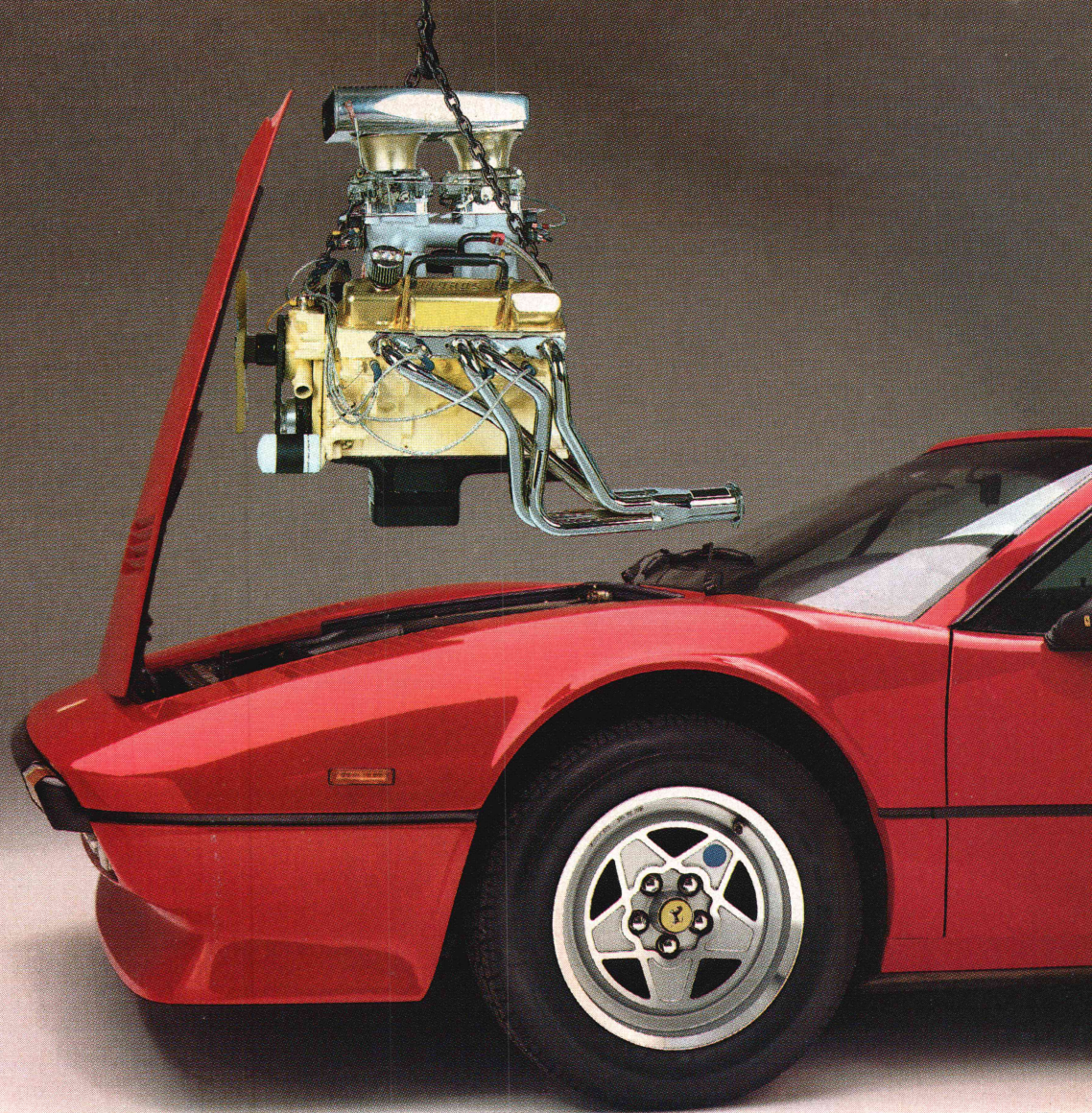
City: _____ Exp. Date: _____

VISA or MC #: _____

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

Watch us!

- ◆ 2034 BLAKE STREET
- ◆ BERKELEY, CA 94704
- ◆ (415) 540-5441



AT™ Pfantasies for your PC or XT.™

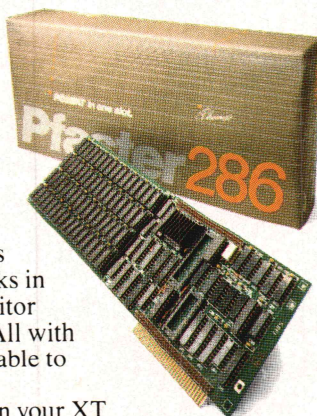
Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle



hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."

If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

Programmers' Pfantasies™
by

Phoenix

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation. For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

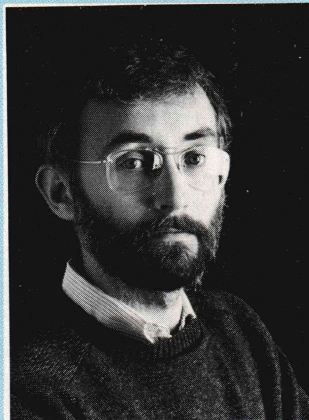
Circle no. 91 on reader service card.

Late 1975 and early 1976 saw a number of new magazines for a new group of readers: the pioneering users of the first microcomputers. Most were heavy on hardware because there was essentially no software to write about back then. In January 1976, a remedy for this situation appeared in the form of a magazine providing the tools needed to develop software: *Dr. Dobb's Journal of Tiny BASIC Calisthenics & Orthodontia*.

Ten years later, *DDJ* is still providing software tools for serious programmers. We've replaced "calisthenics and orthodontia" with other metaphors and have moved upscale from a black-and-white newsletter to a four-color magazine, but we still publish the only pages in which you can find compilers, assemblers, and programming tools reviewed, designed, analyzed, and source-listed, with the whole process watched over by the most knowledgeable readership in the industry.

Was that last paragraph excessively self-congratulatory? Pardon it, but there is occasion for it: This month is *DDJ*'s tenth birthday. This birthday issue features articles on 68000 programming, leading off with Edward Ream's assembler that thinks it's a compiler. In a more overtly festive act, this issue also unveils the new design for *DDJ*.

The new design should more directly indicate what the magazine provides: programming tools in the form of articles and listings, reviews, columns, a forum for discussion of programming issues, and services such as product listings. The title treatment, while harking back to the original cover design, makes it clear that what this magazine contains are software tools. Inside, we've gathered all the contents into five sec-



tions: Articles, Reviews, Columns, Forum, and Programmers' Services. Forum contains the editorial, letters, an editorial cartoon, and an invited viewpoint—this month by 68K authority Hal Hardenberg. Programmers' Services

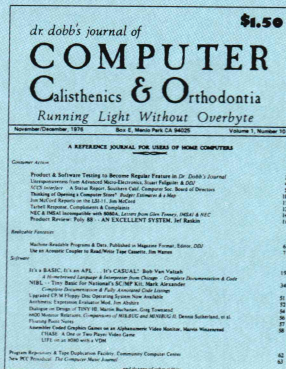
will grow; this month it contains a section of information on the professional side of programming. No room for reviews this month.

We are fortunate to have some excellent columnists who provide a personal view along with useful tips and utilities. Unfortunately, two of them are retiring from their columnist duties: Dave Cortesi and Bob Blum. Bob is just too busy to keep up a monthly column and to do as good a job as he wants; he'll continue to contribute articles and to maintain his CP/M bulletin board. He has written a sign-off letter that appears on page 8. Dave is leaving computer writing for the writing of fiction. Both were longtime contributors and will be missed. We are actively, if grudgingly, seeking replacements. Perhaps you know a candidate?

Finally, there's the electronic component we've added to the magazine. By the time you read this, *DDJ* should be up on CompuServe. See page 17 for details.

Michael Swaine

Michael Swaine



Editorial

Editor-in-Chief Michael Swaine
Managing Editor Vince Leone
Editorial Assistant Sara Noah Ruddy
Technical Editor Allen Holub
Contributing Editors Ray Duncan
 Allen Holub

Copy Editors Laura Kenney
 Rhoda Simmons

Special Projects Editor Frank DeRose

Production

Production Manager Bob Wynne
Art Director Shelley Rae Doeden
Production Assistant Alida Hinton
Typesetter Jean Aring
Cover Design Shelley Rae Doeden

Circulation

Sub. Fulfillment Mgr. Stephanie Barber
Subscription Mgr. Maureen Kaminski
Book Marketing Mgr. Jane Sharninghouse
Single Copy Sales Mgr. Stephanie Barber
Circulation Assistant Kathleen Shay

Administration

Finance Manager Sandra Dunie
Business Manager Betty Trickett
Accounts Payable Supv. Mayda Lopez-Quintana
Accounts Payable Asst. Denise Giannini
Billing Coordinator Laura Di Lazzaro
Accountant Marilyn Brown
Adm. Coordinator Kobi Morgan

Advertising

Advertising Director Shawn Horst (415) 424-0600
Advertising Sales
 Walter Andrzejewski (617) 567-8361
 Lisa Boudreau (415) 424-0600
 Beth Dudas (714) 643-9439
 Michele Beaty (317) 875-8093
Systems Manager Ron Copeland
Administrative Manager Anna Kittleson

M&T Publishing, Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President and Publisher Laird Foshay

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0278-6508**

Customer Service: For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6947. For book, back issue, or disk order problems call 415-424-1474.

Subscription Rates: \$29.97 per year within the United States. Foreign subscription rates: \$56.97 for airmail, \$46.97 for surface mail. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank.

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 686-1520 TEL-EX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.



People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.



The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths ... quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster. Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/Link Time
Dhystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Sensing Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	\$499
Aztec C86-d Developer's System	\$299
Aztec C86-p Personal System	\$199
Aztec C86-a Apprentice System	\$49

All systems are upgradable by paying the difference in price plus \$10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree \$395	Greenleaf \$185
PHACT \$250	PC-lint \$98
HALO \$250	Amber Windows \$59
PRE-C \$395	Windows for C \$195
WindScreen \$149	FirstTime \$295
SunScreen \$99	C Util Lib \$185
PANEL \$295	Plink-86 \$395

MACINTOSH, AMIGA, XENIX, CP/M-68k, 68k ROM

Manx Aztec C68k

"Library handling is very flexible ... documentation is excellent ... the shell a pleasure to work in ... blows away the competition for pure compile speed ... an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C Macro Assembler	Creates Clickable Applications
Overlay Linker	Mouse Enhanced SHELL
Resource Compiler	Easy Access to Mac Toolbox
Debuggers	UNIX Library Functions
Librarian	Terminal Emulator (Source)
Source Editor	Clear Detailed Documentation
MacRam Disk -c	C-Stuff Library
Library Source -c	UniTools (vi,make,diff,grep) -c
	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System	\$499
Aztec C68d-d Developer's System	\$299
Aztec C68k-p Personal System	\$199
C-tree database (source)	\$399
AMIGA, CP/M-68k, 68k UNIX	call

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3	\$399
Aztec C65-d Apple DOS 3.3	\$199
Aztec C65-p Apple Personal system	\$99
Aztec C65-a for learning C	\$49
Aztec C65-c/128 C64, C128, CP/M	\$399

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST. Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX (\$3000), PDP-11 UNIX (\$2000), MS-DOS (\$750), CP/M (\$750), MACINTOSH (\$750), CP/M-68k (\$750), XENIX (\$750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are \$300 to \$500 (non VAX) or \$1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	\$349
Aztec C II-d (CP/M)	\$199
C-tree database (source)	\$399
Aztec C80-c (TRS-80 3 & 4)	\$299
Aztec C80-d (TRS-80 3 & 4)	\$199

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories, Lattice TM Lattice Inc., C-tree TM Faircom, Inc., PHACT TM PHACT ASSOC., CI Optimizing C86 TM Computer Innovations, Inc., MACINTOSH, APPLE TM APPLE, INC., Pre-C, PLINK 86 TM PHOENIX, HALO TM Media Cybernetics, Inc., C-tree, PC-lint TM GIMPLE Software, WindScreen, SunScreen TM Suntec, PANEL TM Roundhill Computer Systems Ltd., WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC., CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE Int.

Circle no. 108 on reader service card.

LETTERS

**Blum Signs Off**

Dear DDJ,

Being a part of DDJ for practically three years has been an incredible experience. Even though I won't be writing a regular column anymore, I don't feel that I am losing a thing. The many new friendships that I now enjoy and the experiences that I shared will last a lifetime.

Of particular importance to me has been your feedback. The many letters I received, even when critical, were a delight to read and helped guide me in preparing material meaningful to you.

Even though I won't be a regular participant in future issues of DDJ, I have a number of CP/M projects under way that I hope will be accepted for publication in the future.

In closing for the last time, allow me to invite you to continue to visit my bulletin board system and to again thank each of you for looking in on the CP/M Exchange each month.

Bob Blum

5536 Colbert Trail

Norcross, GA 30092

Bob Blum's RCP/M system is available for your use 24 hours a day, 7 days a week. Reach it by dialing (404) 449-6588.

Dear DDJ,

I have been searching (with no success) for the address and subscription

cost for *DTACK Grounded*. Would you or any of your readers have this information?

Thank you for your time.

Calvin Dodge

4490 Yukon Ct., #2A

Wheatridge, CO 80033

DTACK Grounded, one of the best newsletters on programming and the industry, ceased publication with Issue 45 to allow its editor, Hal Hardenberg, to devote more time to software development. His company, Digital Acoustics, is healthy and is mailing subscription refunds to subscribers. Back issues are still available from Digital Acoustics, 1415 E. McFadden, Ste. F, Santa Ana, CA 92705.—ed.

Editors

Dear DDJ,

Mark Edwards is to be congratulated for his review of editors in the November 1985 issue of *Dr. Dobb's*. Anyone who ventures into this highly personalized field takes his life in his hands, and to attempt a survey of ten editors is

chutzpah indeed. To do it with taste, relative completeness, and fairness is no mean task.

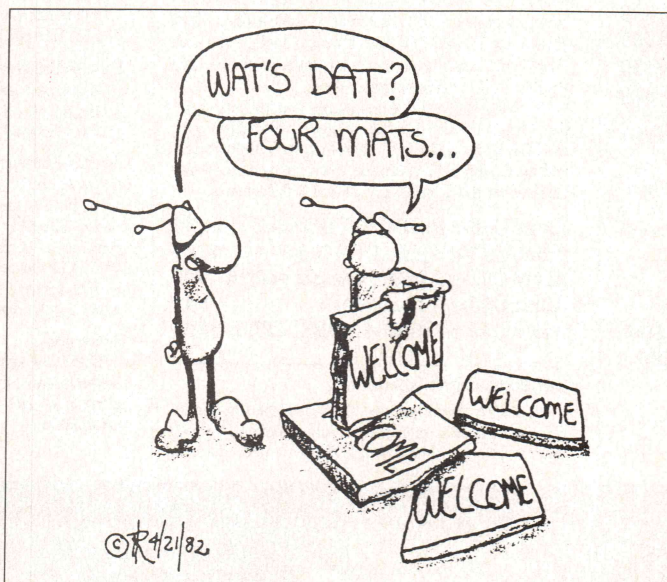
As Mr. Edwards notes, EC is in a high degree of flux, but I would note that my version (probably later than his, given the lead time it requires for an article) can do backward searching and does not suffer the poor error handling he referred to. An important aspect of any software is that of support, and I would just like to comment that I have found the authors of EC superlative in that aspect.

A very nice feature of EC (among many others) is its maintenance of the DOS commands in a buffer. In all my editing I use APX Core, which offers multiple DOS partitions. EC did work with APX but lost the ability to maintain the DOS command buffer under it. I called Gene Brown, the principal author of EC, and informed him of this and also gave him the address of APX. Much to my surprise and delight, I received three (count them,

3) copies of the final version. One was the standard, and the other two were potential solutions to the APX compatibility problem. They all worked, and the solutions did so beautifully. How many companies will give such a response?

I have an additional comment with respect to the relevance of the compatibility of editors with such "enhancers" as SideKick. My experience has been that in the long run the side effects of these programs are less desirable than their offerings. Even more to the point, though, XyWrite, BRIEF, and EC (and probably others in the review also) are sufficiently internally complete that you really do not need such enhancers. BRIEF, with its powerful macro capability, allows you to create these enhancements for the most part, as does XyWrite for those it does not have; EC has most of them built-in, and the few it doesn't, you can create. Communications are accessible by any of these editors because you can exit to DOS, so you have the advantage of no overhead and no potential side effects. With BRIEF, I even wrote a perpetual calendar to mimic that aspect of SideKick.

Because people tend to use editors for short correspondence, a relatively important aspect is their "reformatting" speed as most editors do not (understandably) maintain word wrap for text insertion after a line has been entered. One of the problems I found with EDIX was its intolerably slow reformatting—in



Wendin® puts Power and Punch in your PC

ORDER NOW!
Prices on PCUNIX™ and
PCVMS™ go up to \$99
effective January 1, 1986

Operating System Toolbox™ only \$9900

Operating System Toolbox™ is a software construction set that enables you to build your own multitasking, multiuser operating system. All you need is your creativity and imagination to write a shell and link it with the Toolbox. What you receive for your effort is an operating system that interacts with the way **YOU** work, performs the jobs that **YOU** need in the way **YOU** want them done. Toolbox runs MS-DOS and PC-DOS programs with full source code written in Microsoft "C", organized into seven basic modules, all on disk. The package includes both source and object code for each module (just in case you don't have Microsoft "C"), plus a fantastic, in-depth, step-by-step instruction manual on how to build your personal operating system.

The best part of the news is the price — so low, you'll want to order today, just to keep up with the state of the art in operating systems.

ORDER HOTLINE
509/235-8088
Master Card & Visa

Foreign orders inquire about shipping.
Domestic orders add \$3.50/1st item, \$1.00 each additional item for shipping, handling and insurance.

Washington residents add 7.8% sales tax.



WENDIN®

**BOX 266
CHENEY, WA 99004**

The people who make quality
software tools affordable.

DEALER INQUIRIES WELCOME!

PC-DOS is a trademark of IBM
MS is a trademark of Microsoft
Unix is a trademark of AT&T
VAX/VMS is a trademark of Digital Equipment Corporation

Wendin and XTC are Registered Trademarks of Wendin, Inc.
PCVMS, PCUNIX, and Operating System Toolbox are Trademarks of Wendin, Inc.

PCUNIX™ only \$4900

PCUNIX™ is powerful! It is a true multitasking, multiuser operating system similar to the popular UNIX® operating system from AT&T. This first release includes 30 utilities, designed to make your program development a snap.

PCUNIX™ is versatile! It runs MS-DOS and PC-DOS programs, so that you'll never have to buy another set of development tools! Simply use your existing compilers and linkers. You need nothing more! As with the development of all of our Operating Systems, Wendin has used the Wendin Operating System Toolbox™ to construct PCUNIX™. Because of this, your programs will have access to the over 70 enhanced system services found in the Wendin Operating System Toolbox!™

PCUNIX™ is complete! Full source code to the shell and all utilities, written in "C", is included with this incredible package. And if you would like the source to Wendin's Personal Operating System Kernel, you may also want to purchase the Wendin Operating System Toolbox.™

PCVMS™ only \$4900

PCVMS™ is Wendin's version of the popular VAX/VMS operating system, which was developed by Digital Equipment Corporation for their line of VAX computers. PCVMS™ turns your IBM-PC, XT, AT, or true compatible into a supercharged, multitasking, multiuser workstation that runs MS-DOS and PC-DOS programs.

As with the development of all of our Operating Systems, Wendin has used the Operating System Toolbox™ to construct PCVMS, allowing your programs access to the over 70 enhanced system services available in the Toolbox.

PCVMS™ comes with full source code to the PCVMS™ shell and its utilities, all written in "C". If you also want the source to Wendin's Personal Operating System Kernel, you may want to purchase the Wendin Operating System Toolbox.™

XTC® only \$9900

XTC® is the world's first multitasking editor for the IBM-PC. It also runs on IBM-XT and IBM-AT computers, as well as true compatibles. Designed BY programmers FOR programmers, XTC is the ultimate editing tool for software developers using C, PASCAL, ASSEMBLY, BASIC, FORTRAN, and other languages.

Why is XTC® the ultimate tool for editing in your development environment? Because it has powerful features like MULTITASKING MACROS, WINDOWS, TEXT BUFFERS, UNDO N TIMES, MACRO PROGRAMMING CONTROL STRUCTURES and VARIABLES, as well as blinding speed that leaves other editors in the stone age.

XTC® comes with full source code written in Microsoft Pascal, on 2 DSDD disks.

LETTERS

(Continued from page 8)

this respect VEDIT and EC are lightning fast, and BRIEF (with its free reformat macro) is only slightly faster than EDIX. XyWrite of course does it like any respectable word processor.

I enjoyed reading "Wired Tales" in Dave Cortesi's column in the December issue and offer the following. I recently decided to upgrade my PC with a larger power supply (64 watts to 135 watts) and, being relatively frugal, ordered one from a mail-order house for \$89. When it arrived I (foolishly) started to install it: removed my old power supply, put the new one in, and started to make the connections. To my dismay, it turned out that the connectors were poorly constructed and I could not get a good fit. I decided to return the supply for credit, not wanting to chance a repeat performance. It took me about \$15 worth of phone calls to finally talk to someone who was authorized to issue an RA, and that same person informed me that there was a 10 percent re-

stocking fee. Adding in the cost of the UPS shipment back, I wound up paying \$30 for some futile labor in taking out and replacing my power supply, losing a nontrivial amount of time in the process.

Finally, I offer a poser. I recently purchased an uninterruptible power supply (300 watts) from Qubie. When it was delivered, I tested it in the usual way—I pulled the power plug—and it worked fine. Just to be doubly sure, I also opened the main circuit breaker in the house, and the same excellent performance repeated. A few days ago, we had our first real power outage while I was working at my PC, and much to my surprise, I lost all my work. When the power returned about one minute later, I immediately made the "pull the plug" test and it worked. I called Qubie's technical department, and no one could offer a solution, but the company did offer to send me a replacement (which I accepted). In the meantime I have been thinking about the possible causes of the loss of my work and believe I may understand why it hap-

pened, but I would love to hear from others.

Morton F. Kaplon
11 White Birch Dr.
Pomona, NY 10970

Dear DDJ,

Although I greatly enjoyed Mark Edwards' editor review (DDJ, November 1985), I would like to make a correction or two and point out a problem with this kind of review. I think Mr. Edwards did a terrific job given the difficulty of trying to learn about ten text editors, much less trying to comment intelligently about them all. I use Pmate (Version 3.37) on a daily basis, however, and I am familiar enough with it to have caught a couple of errors in the review.

The first (and smallest) is that Pmate does indeed allow you to undo the deletion of a single character. What it will not allow is the retrieval of a character you have backspaced away. On a PC, this is the difference between the Delete key and the ← (backspace) key. The difference can be annoying, but it is frequently more helpful than not once you are aware that Pmate works in this way.

A more serious error is in Mr. Edwards' Pmate macro to count braces. The macro he presents does indeed work, but it is terribly slow. Trying it on a sample C program out of one of my working directories (on a Compaq Plus), it ran for more than six minutes before I aborted it—unfinished. There is a much better Pmate macro for brace counting, which I present in Table 1, page 10. This brace-counting macro finished the same piece of code as above in less than six seconds (and is slowed slightly because it is constructed to work with a file

of any size as opposed to one entirely in memory). This is a huge difference from the results reported in the review. Because of differences in the test files, it is impossible to say whether this performance is better or worse than BRIEF or EMACS, but it is certainly of the same order.

It is worth noting that the macro presented in Table 1 works in much the same manner as the BRIEF and EMACS macros presented by Mr. Edwards. In particular, it allows the editor itself to search for the characters to be counted rather than stepping through the file in a clumsy character-by-character fashion. I suspect that a properly rewritten macro for VEDIT PLUS would show the same kind of dramatic speed improvement. There might even be some hope for XTC if the proper macro formulation were used.

This in turn points out a serious difficulty in benchmarking editors. A user who is familiar with an editor will undoubtedly be able to generate a better benchmark than one who isn't. I would never have even considered writing a brace-counting program for Pmate in the way Mr. Edwards did. The way Pmate works guarantees that his macro is among the slowest possible for the task.

Based on my experience, I would tend to agree with Mr. Edwards' comments about the functionality and features of each editor he reviewed. Based on my findings for Pmate, though, I would be inclined to disregard the entire benchmark table except for items A, B, F, and maybe C.

Mr. Edwards has certainly proved that it is im-

```

;***
;*** Pmate brace counting macro program
;***
0v1      ;set variable 1 to 0
ua       ;go to top of file
[        ;begin loop
  e       ;suppress error messages
  us(     ;search for an open brace
  @t=0_   ;if none found, exit loop
  va1     ;else increment count of braces
]        ;end loop
ua       ;return to top of file
[        ;begin loop
  e       ;supress error messages
  us)     ;search for closing braces
  @t=0_   ;if none found, exit loop
  -1va1   ;else decrement count of braces
]        ;end loop
b2e      ;go to buffer 2
@1\      ;put the result in buffer 2

```

Table 1

Mark Williams

Now the biggest name
in C compilers comes in a size
everybody can afford.
Let's C.™

Introducing Mark Williams' \$75 C compiler. Want to explore C programming for the first time? Or just on your own time? Now you can do it in a big way without spending that way. With Let's C.

This is no little beginner's model. Let's C is a powerful programming tool, packed with all the essentials of the famous Mark Williams C Programming System. The one chosen by Intel, DEC, Wang and thousands of professional programmers. The one that wins the benchmarks and the reviewers' praise:

Mark Williams Let's C

- For the IBM-PC and MS-DOS
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX™ compatibility and complete libraries
- Small memory model
- Many powerful utilities including linker, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source
- Supported by dozens of third party libraries
- Upgradeable to C Programming System for large scale applications development

Let's C Benchmark Done on an IBM-PC/XT, no 8087.
Program: Floating Point from BYTE, August, 1983.

Exec Time in Seconds

Let's C	134.20
MS 3.0	347.45

"(This compiler) has the most professional feel of any package we tested..."—BYTE
"Of all the compilers reviewed, (it) would be my first choice for product development."—David W. Smith, PC WORLD

And now for more big news. Get our revolutionary csd C Source Debugger for just \$75, too. You can breeze through debugging at the C source level ignoring clunky assembler code.

Affordable, powerful, debuggable. Mark Williams Let's C is the big name C compiler at a price you can handle. Get your hands on it now.

Use this coupon or charge by calling toll-free:
1-800-MWC-1700. In Ill. call 312-472-6659.

Mark Williams Let's C

\$75

Please send me:

_____ copies of Let's C and _____ copies of csd (C Source Debugger) at \$75 each. (Ill. residents add 7% sales tax.)

☐ Check ☐ Money Order ☐ Visa, MasterCard or American Express

Name _____

Address _____

City _____ State _____ Zip _____

Card # _____ Exp. Date _____

Signature _____



**Mark
Williams
Company**

1430 West Wrightwood
Chicago, Illinois 60614

LETTERS

(Continued from page 10)

possible for a single person to learn enough about ten different editors to be able to compare each at its best.

Brad Chase
P.O. Box 705
Exeter, NH 03883

Modula-2

Dear DDJ,

While perusing your otherwise excellent November issue, I felt a sudden stab of pain. Looking closer, I saw the source of my discomfort was "Bit Manipulation in Modula-2." I feel an overwhelming need to comment.

"I knew from the outset that it was impossible to match C's simplicity." This tells me that the author is primarily a C hacker. Modula-2 is perfectly capable of matching this "simplicity," and in exactly the same manner. (See Table 2, below.)

The other operations are left as exercises for the reader.

Lawrence C. Smith
51 Lake St.
Nashua, NH 03060

C Compilers

Dear DDJ,

I love Dr. Dobb's and have subscribed for years. I notice a trend away from assembly, CP/M, 8-bit, and the less wealthy user, though. In particular, your issue on C language was a

disappointment. The following are my reasons, along with some suggestions. I understand your problems in appealing to a divergent group, however, and appreciate your willingness to listen to all our suggestions.

1. The main article on C comparisons told me more than I really cared to know about the languages. I would have trusted the writers if they had summarized their results. An article this long and detailed should have a summary at the beginning.

2. The article did not state which of the languages examined were available on either S-100 machines, on CP/M, or on 8-bit machines.

3. I did not see mention of several of the Cs I use—BDS C, Small-C, or Tiny C or any mention of Lifeline's products—and that would have been important to me. I am sure other Cs could have been found as well—so how much longer would it have taken to wait a bit and do a truly definitive review?

Frederic Schlamp
2205 Meadowview Rd.
Sacramento, CA 95832

The comparative review of C compilers in our August 1985 issue was strictly dedicated to MS DOS-compatible compilers. We will review C compilers for other environments, including CP/M,

in 1986.—ed.

Columns

Dear DDJ,

In your July 1985 issue the 16-Bit Software Toolbox column began with the line "One of the most novel features added in Version 2 of MS DOS is the concept of 'installable device drivers.'" I would like to say that this concept may be new and novel for Microsoft and MS DOS, but it is certainly not a new and novel concept for other operating systems available for microprocessors.

The OS-9 operating system has had the concept of user-installable device drivers since its initial 6809 Level 1 release in 1978. In fact, OS-9 is totally modular in nature and allows the user to add new device descriptors, device drivers, and new file managers if required. In addition to supporting installable drivers, OS-9 has included the "novel" MS DOS concepts of hierarchical director structures and pipes. OS-9 also gives full support to I/O redirection, multiprocessing, and multitasking—concepts much more akin to Unix.

OS-9 may not be as well known as MS DOS is, but it does have a large and rapidly growing following in the 6809 and 68XXX world today. MS DOS has added nothing novel to its OS; it is adding features that are expected and required for an OS in today's world. These features have been around in OS-9 and OS-9/68000 for some time.

Tim Harris
Microware Systems Corp.
1866 N.W. 114th St.
Des Moines, IA 50322

Dear DDJ,

"It isn't what you don't know that hurts you, it's

what you know that ain't so." In the September Of Interest column, the author states that APL on the IBM PC requires an 8087. This is only true for IBM's own APL, not for the other five (STSC, Sharp, Portable Software, Watcom, and NIAL Systems). Most of them run on the PCjr.

Edward M. Cherlin
6611 Linville Dr.
Weed, CA 96094

Dear DDJ,

I was interested to read the portion of your Dr. Dobb's Clinic (October 1985) regarding the intricacies of manipulating path names. Some time ago, we at POLYTRON also had the pleasure of figuring out how to do exactly what you discussed. Our solution, which produced some functions that are more general purpose, might interest your readers. The primary components of our solution are:

1. A function to determine if a given path name represents a directory—that is, it is a drive ID or an actual director but not a normal file:

```
int is_dir(name)
char *name;
```

2. A function to determine if a given file name contains wildcard characters:

```
int is_wild(name)
char *name;
```

3. A general-purpose function for generating a new file name given an existing file name, an optional new extension, and an optional new path:

```
char *
bld_fname(pathp,
namep, extp)
char *pathp;
```

```
(* assume ch:=CHAR(255) *)           (* clearing bits *)
(* clear bit 7 *);
ch:=CHAR(BOOLEAN(ch) & BOOLEAN(07FH));
(* result is ch:=CHAR(127) *)
(* assume ch:=CHAR(15) *)             (* setting bits *)
(* set bit 5 *);
ch:=CHAR(BOOLEAN(ch) OR BOOLEAN(10H));
(* result is ch:=CHAR(63) *)
```

Table 2

Turbo, who?

Do you have to give up power and advanced potential to get ease of use and affordability? Not anymore. Because now, you can have **UCSD Pascal** for only \$79.95!

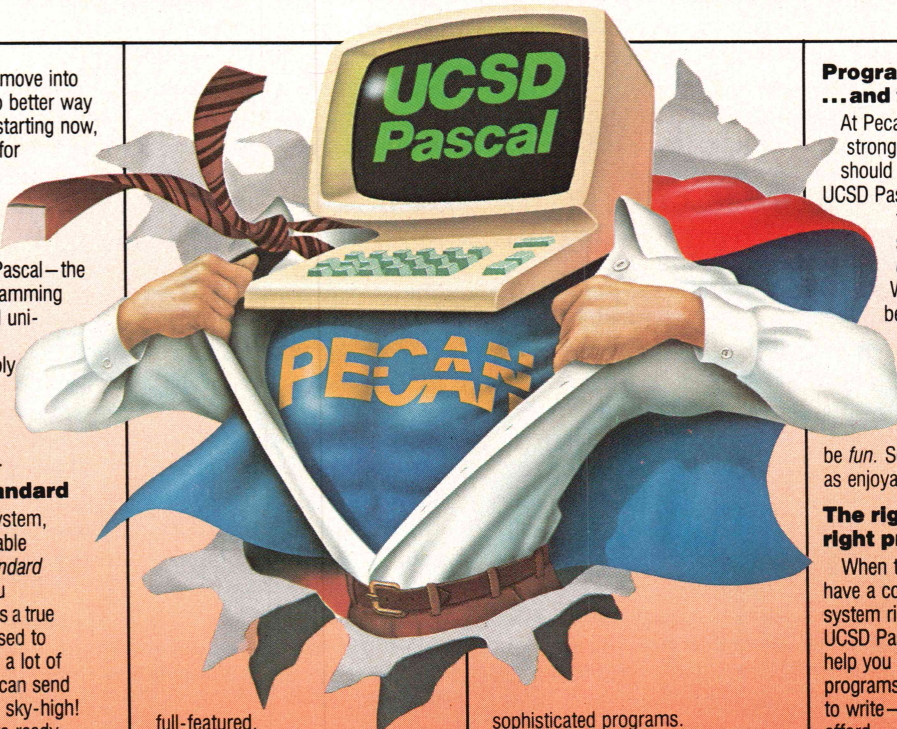
If you're making your move into programming, there's no better way to go than Pascal. And starting now, you *don't* have to settle for a stripped-down version of Pascal in order to get a price that's right. Instead, you can choose UCSD Pascal—the recognized Pascal programming standard in colleges and universities throughout the country—at the incredibly low introductory price of \$79.95 for your PC-DOS, MS-DOS, or other popular computer.

Start with the standard

With an entry-level system, you spend a lot of valuable time learning a non-standard form of Pascal. And you don't get all the capabilities a true Pascal system is supposed to deliver—unless you buy a lot of add-on utilities—which can send the cost of your system sky-high! Worst of all, when you're ready to tackle anything more than short, simple programs—you have no choice but to move up to a more sophisticated system (like UCSD Pascal). And at that point, you also have to *relearn* standard Pascal.

UCSD Pascal has everything you need

With UCSD Pascal, you get a



full-featured, professional programming tool that's being used right now in the development of major scientific and business applications. The system comes with an outstanding text editor, a complete on-line tutorial, 8087 math coprocessor support and BCD (decimal arithmetic) included in the package at no extra cost. In fact, UCSD Pascal contains virtually everything you need—as standard equipment—for developing the simplest to the most

sophisticated programs.

UCSD Pascal is available for MS-DOS, PC-DOS, UNIX, VMS, MSX and many other operating systems. You can use UCSD Pascal to write programs of any size on virtually any computer, and port them to any other computer. And if speed is what you're after, the latest native code version of UCSD Pascal actually benchmarks favorably with Turbo Pascal® in execution time!

Programming that's easy ...and fun!

At Pecan Software Systems, we strongly believe programming should be as easy as possible. UCSD Pascal was originally designed for teaching programming skills, so it's extremely easy to learn and to use. With UCSD Pascal, you'll be developing programs right from the start that are easy to write, easy to understand, and easy to maintain. We also believe that programming should be fun. So we've made UCSD Pascal as enjoyable to use as it is powerful.

The right tool at the right price

When the fun gets serious, you'll have a comprehensive programming system right at your fingertips with UCSD Pascal—a system that will help you develop those big-league programs you may eventually want to write—at a price you can readily afford.

Put UCSD Pascal programming power on your PC now for only \$79.95! Order by mail today or phone now 1-800-63-PECAN. UCSD Pascal—the original standard of Pascal programming excellence. The new leader in Pascal price/performance.

PECAN

The UCSD Pascal Company
Pecan Software Systems, Inc.
1410 39th Street, Brooklyn, NY 11218
718-851-3100

UCSD Pascal

UCSD Pascal \$79.95 (for PC-DOS, MS-DOS, AMIGA, APPLE, ATARI 520, MACINTOSH, RAINBOW, TANDY, as well as most popular 8/16/32-bit systems).

Price includes 8087 support and BCD. 8/16/32-bit systems.

Please send me _____ copies of UCSD Pascal for my _____ (Operating system)

(Name and model of computer) _____
My disk size is 3 1/2" _____ 5 1/4" _____ 8" _____
Total amount (NYS add appropriate tax) _____

Payment by ☐ VISA ☐ MC ☐ US Bank Check ☐ Bank Draft

Card Number: _____

Credit Card Expiration Date: _____

Mail to: Pecan Software Systems, Inc.
1410 39th Street
Brooklyn, NY 11218
ITT Telex No. 494 8910
CompuServe Code 76703, 500

Name _____

Shipping Address _____

City _____

Telephone _____

UCSD Pascal
Not copy protected
60-day money-back guarantee

CREDIT CARD ORDERS
CALL TOLL-FREE
1-800-63-PECAN
(NYS) 1-800-45-PECAN

Call or write for UNIX, VAX or other UCSD Pascal versions—and ask about our powerful Pascal add-ons, too. **SCHOOLS:** Contact us for our special educational discounts!
Call toll-free or enclose a check with this coupon to place an order. Please add \$2.50 for shipping within the US. Foreign orders add \$10 and make payment by bank draft payable in US dollars on US bank. New York State residents add appropriate sales tax.

AMIGA is a trademark of Commodore Electronics LTD.

APPLE & MACINTOSH are trademarks of Apple Computer.

ATARI 520 is a trademark of Atari Corporation.

RAINBOW is a trademark of Digital Equipment Corporation.

TANDY is a trademark of Radio Shack.

Turbo Pascal is a registered trademark of Borland International.

LETTERS

(Continued from page 12)

```
/* the new path */
char *namep;
/* the original name */
char *extp;
/* the new extension */
```

This returns a pointer to allocated memory containing the newly constructed file name. The original name may or may not have a path or extension. If either *pathp* or *extp* point to a null string, the respective component of the original name is omitted from the new name. If either *pathp* or *extp* is a null pointer, the respective component of the original is used in the new name.

4. A wildcard file name expansion function with optional ability to prepend the path of the wildcard name is shown in Table 3 (below). This returns a pointer to a linked list of names matching the wildcard name. Each node of the list is contained in sep-

arately allocated memory and is of sufficient length to contain the matching name beginning at *name[0]*.

5. Functions to extract each of the three components of a file name—the path, the root, and the extension are shown in Table 4 (below). These functions each return a pointer to allocated memory containing the respective component of a file name.

Using these functions, you can do just about anything you wish with file names. In fact, they are used in nearly all our products and continue to be used in new development work.

These functions, along with a host of others, are available in the POLYTRON C Library 1. This package contains routines that function under DOS 1.x, DOS 2.x and later, and both/either. The functions are provided in linkable form (libraries), as well as in full source form (C and assembler) for the IBM PC/

XT/AT and compatibles.

Donald K. Kinzer
POLYTRON Corp.
P.O. Box 787
Hillsboro, OR 97123

Dear DDJ,

Here is a report on a software company for the benefit of prospective buyers. I purchased a cross-assembler from 2500AD in November 1984. What they sent me was a nonfunctioning program. Soon after receiving the program, I sent two letters, made two phone calls, and finally spoke to someone who said the company would send me a good copy when it had fixed the program. It has been a year, the firm still has my \$200, and I have not yet received a functional copy of the program. Even JRT Pascal was functional, and look at the price difference!

For the record, here are the defects that I have found (so far):

1. The assembler and the linker are incompatible. The linker makes the wrong assumptions about the relative order of least/most significant bytes, which makes it impossible to link object modules. The only way to get an executable file is to put all your code into one module and use absolute addressing (with an ORG statement).
2. The assembler gives the wrong machine code for several of the opcodes. This makes for difficult debugging.
3. Some of the pseudops listed in the manual don't exist in the actual assembler.
4. Some of the pseudops that do exist do not work (at least one).
5. The assembler has no provision for allowing the programmer to specify the short forms of the relative

jump and call instructions.

6. The symbol table that is printed at the end of the listing usually contains sections of garbled mess.

7. The printed symbol table lists all intermediate values of labels that are redefined many times with the DEFL statement.

8. Some errors in the source code cause the assembler to crash with no error message.

9. The linker crashes with a nonsense error message under some conditions that seem to have something to do with certain exact lengths of files.

Neil R. Koozer
Kellogg Star Rt.
Box 125
Oakland, OR 97462

In the September issue, we ran a comparative review of PC TEX and MicroTEX. At that time only PC TEX included INITEX, but we stated that MicroTEX was also scheduled to include INITEX in an August update. As of the time this issue went to press, MicroTEX still does not include full INITEX capabilities. We have been informed by Addison-Wesley that a new version of MicroTEX, including INITEX, will not be available until January 1, 1986. Check with Addison-Wesley before placing an order. Both Addison-Wesley and Personal TEX are already distributing the Textset laser printer and screen preview driver.—ed.

DDJ

```
/* element of a linked list of matching names */
struct file_list
{
    struct file_list *next; /* ptr to next node */
    char name[1]; /* a file name */
};
struct file_list *
expand(wildname, add_path)
char *wildname; /* the name to match */
int add_path; /* non_zero if path of wildname should be
                prepended to matching names */
```

Table 3

```
char *
path_of(name)
char *name; /* the name from which to extract the path */
char *
root_of(name)
char *name; /* the name from which to extract the root */
char *
ext_of(name)
char *name; /* the name from which to extract the
              extension */
```

Table 4

Another in a series of
productivity notes on
MS-DOS™ software
from UniPress.

Subject: Multi-window full screen editor.

Multiple windows allow several files (or portions of the same file) to be edited simultaneously. Programmable through macros and the built-in compiled MLISP™ extension language.

Features:

- Famed Gosling version.
- Extensible through macros and the built-in compiled MLISP extension language.
- Dozens of source code MLISP functions; including C, Pascal, LISP and MLISP syntax checking.
- EDT and simple WordStar™ emulation modes.
- MS-DOS commands can be executed with output placed in an EMACS window.
- Run a compile on your program and EMACS will point to any errors for ease of debugging.
- EMACS runs on the IBM-PC™ (XT/AT), TI-PC™, DEC Rainbow 100+™, HP-150™ or any other MS-DOS machine. Requires at least 384K.

Price:

EMACS binary \$325
EMACS source 995
One month trial 75

Also available for UNIX™ and VMS™
Call for pricing.

TEXT EDITING

UNIPRESS EMACS™

Subject: Compiler for MS-DOS

Lattice C Compiler is regarded as the finest compiler for MS-DOS and produces very efficient and compact code.

Features:

- Runs on the IBM-PC under MS-DOS 1.0, 2.0 or 3.0.
- Produces highly optimized code.
- Small, medium, compact, and large address space models available.
- Full C language and standard library.
- 8087™ floating point support.
- PLINK™ linkage editor is optionally available to support modular programming.

Price:

Lattice C Compiler \$425
PLINK 395

COMPILER FOR THE 8086™ FAMILY

LATTICE® C COMPILER

Subject: UNIX-like "make" facility now for MS-DOS.

Ps-Make is a powerful programming aid providing time saving steps for the programmer by compiling only changed components of a program.

Features:

- Ps-Make compiles only changed components of your program. There is no need to execute long batch files that re-compile your entire system.
- Ps-Make directly executes the compiler, linker and other utilities, or automatically generates a batch command file containing only those commands that must be executed to bring the program up-to-date.
- Uses standard UNIX "makefile" syntax.
- Ps-Make can be used with any compiler, including Lattice C.
- Two modes: Batch requires 64K, Direct requires 256K.

Price:

Ps-Make \$90

For our **Free Catalogue** and more information on these and other software products, call or write:
UniPress Software, Inc.,
2025 Lincoln Hwy., Edison, NJ 08817.
Telephone: (201) 985-8000.
Order Desk: (800) 222-0550
(Outside NJ). Telex 709418.
European Distributor: Modulator SA,
Switzerland Telephone: 41 31 59 22 22,
Telex: 911859

OEM terms available.
Mastercard/Visa accepted.

See our ad in the next issue for
more MS-DOS products, including
PHACT™ and PCworks™.

UNIX "MAKE" FACILITY

PS-MAKE

Trademarks of UniPress: EMACS/MLISP UniPress Software, Inc. MS-DOS
Microsoft, IBM PC, IBM Corp. WordStar, MicroPro Intl. Corp. TIPC, Texas In-
struments, UNIX, APT Bell Laboratories VMS/DEC Rainbow 100+, Digital Equip-
ment Corp., Lattice, Lattice, Inc., HP-150, Hewlett Packard Co., PLINK, Phoenix
Computer Products, Corp. 8086/8087 Intel Corp., PHACT, PHACT Associates, PC
works, Touchstone Software Corp.

UniPress Software

Inefficient C

This column is adapted from DTACK Grounded, The Journal of Simple 68000/32801 Systems, Issue 42.

It has been apparent to me for two or three years now that complex programs (as opposed to the famous but simplistic "hello world" type) written in C consistently run a great deal slower than the same complex programs written in assembly. Examples of this rule can readily be found in the personal computer mass marketplace.

It is true that, in the last year or two, many intelligent and experienced programmers have asserted that C has little or no high-level language (HLL) overhead.

There is an apparent conflict here.

Consider the complex problem of writing a BASIC interpreter. You can break this problem down into a large number of simple problems that can then be solved in assembly or, say, C. The speed with which each of the simple problems can be solved depends on how good a match can be made between the problem and the available control constructs, data types, and so on. In some cases the match with the constructs and data types available in

be solved as quickly (or nearly so) in C as in assembly.

So, the C supporters have no shortage of real-world examples of simple problems that can be solved as quickly (or nearly so) in C as in assembly, which leads them to claim that C has no high-level overhead.

Some problems, though, can be solved easily and quickly in assembly and are real bears in C, given the restricted range of operations, constructs, and data types available to C programmers. The large number of simple problems that comprise a BASIC interpreter will include some C-easy problems, some C-not-so-easy problems, and some C-bears. Therefore, any full-function BASIC interpreter written in C will always be a great deal slower than the same interpreter written in assembly.

You need not look for complex ways to analyze the HLL overhead of C (or any high-level language). The fact is, all high-level languages greatly restrict the range of operations, control constructs, and data types available, compared to assembly language. Thus, HLL programmers have a limited number of tools available for solving the large number of simple problems that comprise any solvable complex problem.

Obviously, the programmer with the most complete set of tools can always produce the fastest code. Real-world evidence such as Microsoft's MBASIC (written in assembly) vs. DRI's Personal BASIC (written in C) confirms this rule.

Why, then, do intelligent, experienced programmers claim that C has little or no HLL overhead when abundant real-world evidence contradicts such an assertion?

Well, suppose you have solved a complex problem using C. Then, keeping the 90-10 rule in mind, you go looking for ways to speed up the software. Do you rewrite the program from scratch in assembly, using the greater variety of available operations, data structures, and control constructs? No, you continue to use the data structures and algorithms that you developed in C. That is, you implicitly restrict yourself to the smaller toolkit that is already used by C! Not surprisingly, the resultant program is not much faster than the original. Hence you announce, "Hey, guys, I tried optimizing the problem using assembly and got little or no improvement over the original C. Obviously C has little or no high-level overhead."

Obviously.

HLL inefficiency is acceptable in many environments. Where HLL inefficiency is not acceptable is in the personal computer mass marketplace wherever an efficient alternative is available. DRI's Personal BASIC is essentially a dead issue, being highly uncompetitive with Microsoft's assembly-based BASICs in terms of speed. In turn, the marketplace is widely avoiding Microsoft's C-written FORTRAN compiler.

The individuals who make up the mass marketplace and who vote with their wallets don't care

how hard it was to produce a program or how long it took. They don't care about source documentation or maintainability because they never get to see the source documentation and somebody else maintains the code. They vote for those programs that work both swiftly and well (Lotus 1-2-3, original WordStar, MBASIC, or GW-BASIC) rather than those that run well but slowly (Context MBA, WordStar 2000, or DRI Personal BASIC).

The folks who assert that C has little or no HLL overhead are either talking about simple problems for which C is in fact well suited, or they are observing that, if algorithms written in C are duplicated in assembly, then the assembly versions have little advantage. To reason that C therefore has little HLL overhead is faulty logic.

Other things being equal, the more complete toolbox is always going to win provided the workman knows how to use those tools. That's why good, experienced assembly programmers cost more than HLL coders.

DDJ

by Hal Hardenberg

C is as good as the best that can be devised using assembly, and you have zero HLL overhead. In fact, many simple problems can

DDJ ONLINE

DDJ Goes Online

On January 1, 1986, the Electronic Edition of *Dr. Dobb's Journal* will appear on CompuServe. Through the Electronic Edition DDJ will offer the following:

- We will make available in our data libraries all the listings that appear in the articles and columns of *Dr. Dobb's Journal* every month. We will also offer selections from some of our more popular back issues and run articles and listings that have not appeared in the magazine.
- We will maintain a display area where CompuServe users can read ab-

stracts of the material in the current issue. The display area will also contain general magazine information, such as our editorial calendar, writers' guidelines, and information about advertising rates.

- We will compile comprehensive lists of the commercial software development tools available to microcomputer programmers and feature comparative reviews of products in selected categories. We will also compile selected bibliographies and give capsule reviews of newly released books.
- We will provide a messaging facility with columnists and other SIG members.
- We will stage regular on-

line teleconferences with authors, columnists, and other distinguished guests.

- We will take subscriptions for *Dr. Dobb's Journal*, the magazine. We will also provide a way for readers to register circulation complaints.
- We will provide complete information about other DDJ publications, such as back issues, bound volumes, indexes, *Dr. Dobb's Books*, and *Dr. Dobb's Software*. We will also take orders for these items.

Please watch this department for further announcements about conference schedules or the availability of listings from

back issues. Also, if you have a request for a listing from a back issue, drop us a note or just leave a message in the DDJ Electronic Edition on CompuServe. If we get enough requests for a specific listing, we will try to make it available.

Some of these services may not be available January 1. Most will be free, but some will involve a small charge.

You access the DDJ Electronic Edition by typing go DDJ at any CompuServe system prompt. We hope you will drop by and have a look. See you soon!

DDJ

"XTREE™ has redefined my relationship with my hard drive."

Why XTREE is the New Standard For File and Directory Management.

—PC MAGAZINE

XTREE simplifies file and directory handling by providing single keystroke commands to access, delete, rename, view, move, list or show all files within any and all directories. XTREE displays a graphic picture of your directory organization, instantly shows all the files in each directory or all files across all directories! Just point with the arrow keys (full scrolling and paging in all windows) and press a key. Your menu is clearly displayed at all times.

10 Reasons To Buy XTREE

1. Shows **ALL** files or groups of files in **ALL** directories in one sorted display.
2. Copy, delete or rename multiple files in different directories in **ONE** operation.
3. Automatically copies groups of files across several diskettes.
4. **NOT** copy-protected.
5. Demo available for only \$5.00 includes reusable diskette case (holds up to 5 disks).
6. "I finally found a package that I use routinely to clean up my disks. I recommend it highly. It's a steal at \$49.95."
7. "All XTREE operations execute very quickly. XTREE is a great utility for getting an overall feel of your hard disk. And, if you feel the need to totally reorganize your hard disk, XTREE is unparalleled."

— John Dvorak, Infoworld

— Garry Ray, PC Week

8. "If the program were not already subtitled *The new standard for file and directory management*, I would have described this wonderful new utility the exact same way."

— Phil Wiswell, PC Magazine

9. Only \$49.95 plus \$2 shipping.



10. To Order call 800-634-5545 In CA (818) 990-3457 or send check or money order to:
EXECUTIVE SYSTEMS, INC.
15300 Ventura Blvd., Suite 305, Sherman Oaks, CA 91403

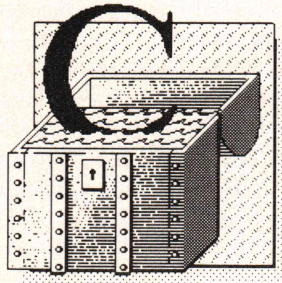
Path: \SPRINGS\BUDGET\1985

<ul style="list-style-type: none"> ROOT SPRINGS BUDGET 1984 1985 1986 TOOLS WORDPROC CONTRACT LETTER 	<p>FILE: *.*</p> <p>DISK: B: XTREE(C:\ES)</p> <p>Available Bytes: 250,000</p> <p>DISK Statistics</p> <p>Total</p> <p>Files: 96</p> <p>Bytes: 61,408</p> <p>Matching</p> <p>Files: 96</p> <p>Bytes: 61,408</p> <p>Tagged</p> <p>Files: 0</p> <p>Bytes: 0</p> <p>Current Directory</p> <p>1985</p> <p>Files: 12</p>
--	---

DIR Available Delete Files Log disk Makedir Rename Show all files
COMMANDS Tag Untag Volume *Tag *Untag
F1 select directory RETURN file commands F1 quit F2 help

C CHEST

A Unix-like Shell for MS DOS



Last month we presented a few support routines, part of a Unix-like shell for MS DOS. This month we're going to continue with the shell itself, describing how it works on a high level. Next month we'll look at it at a lower, functional level. The code itself, because it's so long, will be spread over the next three months.

The shell described here includes functions of the Unix C shell that I use most often, such as aliases, history, and command-line wildcard expansion (more on all this in a moment). It has batch-file capability and will permit nested batch-file execution (unlike MS DOS, which lets you chain from one batch file to another but won't let you return to the original batch file).

It supports a 2,048-byte command line with interactive editing. The long command line is passed to an executed program through an environment variable.

Using the Shell

Commands are entered from the command line, just as in DOS. (Note that \ is a special character to the shell, so use slash (/) or \ \ to separate directory names.) DOS wildcard characters (* and ?) are expanded before a command is executed. So if you say *echo *.c*, the *.c will be expanded to the names of all files having a .c extension before *echo* is invoked. Expanded names are sorted. Several semicolon-delimited commands may be executed from a single command line. For example:

```
cd foo ; pwd ; ls
```

by Allen Holub

changes the current directory to *foo*, prints the full path name, and then prints a list of the files in the current directory.

Command-line editing (as de-

scribed last month) is supported. To summarize:

Cursors—moves the cursor.

Home—gets to the beginning of the line.

End—gets to the end of the line.

Ctrl-right arrow and Ctrl-left arrow—get to the next and the previous word, respectively.

^H—is a destructive backspace.

Del—deletes the character under the cursor. (Typed characters will be inserted at the current cursor location, moving all characters to the left of the cursor over one notch.)

^X—deletes the entire line.

Esc—does the same and aborts.

Return—causes the commands to be executed.

There are several built-in commands:

alias—creates, modifies, or prints aliases (see below). There are two syntaxes:

```
alias
alias name <val>
```

The first prints all currently defined aliases, and the second creates an alias for *name* with the indicated value. <Val> may be anything on the command line, but you have to escape (precede with a \) any character that's special to the shell (or surround <val> with double quotes).

cd—changes a directory or disk:

```
cd foo—changes to the subdirec-
tory foo.
```

cd..—changes to the parent directory.

cd a:—changes to the current directory on the a: drive.

cd a:/foo—changes to the /foo directory on the a: drive.

Cd must be used to change disks, although you can *alias a:* to *cd a:* if you like. The shell checks to see if a disk is in the indicated drive before the drive is logged on.

exit—terminates the shell. Either *exit* or *logout* must be used to leave the outermost shell. Subshells can be terminated with a ^C.

history—prints the history list (see below).

logout—like *exit*, terminates the shell; however, the file /logout.bat is executed before exiting.

pwd—prints out the current working directory (same as *chdir* with no arguments under DOS).

rem—does nothing. May take arguments (which will be ignored). This command is here only for DOS compatibility. The preferable method for commenting a batch file is to start comment lines with a # in the far left column. Note that *rem* is interpreted as if it were a command; that is, the line is put into the history list. Moreover, if a line starts with a *rem* but has a semicolon on it as well, text up to the semicolon will be ignored but the semicolon will be treated normally, and any text following the semicolon will be treated as a second command and executed.

set—creates, modifies, or prints a shell variable (see below). Its syntax is:

```
set
set name [=][value]
```

The first form prints all current shell variables, and the second creates or modifies an existing variable. Both the equals sign and the value fields are optional. If

you omit the value, the alias will expand to a null string.
setenv—changes or creates an environment variable. Its syntax is:

```
setenv name [=][value]
```

Both the equals sign and the value field are optional. An example:

```
setenv PROMPT $p>
```

sets the prompt to the current directory name followed by **>**. This command is very similar to the DOS **set** command. However, **setenv** with no arguments doesn't print the environment. Also, the equals sign is optional.
shift—shifts all the **\$<num>** arguments in a batch file left one notch. For example, if a batch file (**foo.bat**) consisting of:

```
echo $0 $1 $2
shift
echo $0 $1 $2
shift
echo $0 $1 $2
```

is executed with the line:

```
foo first second
```

the following output will be sent to the screen:

```
echo first second
first second
second
```

(**\$0** expands to the file name.)

unalias—removes an alias. The syntax is **unalias name**.

unset—removes a shell variable. The syntax is **unset name**. The default variables (**verbose**, **echo**, **arg**) can't be removed.

History

The history functions let you examine, edit, and reexecute previous commands. The 32 most recently typed commands are remembered in a history list, and each command has a history number associated with it. Once the history list is full, the oldest command is discarded every time a new command is added to the list. The easiest way to see how this mechanism works is with an example. Assume that you've typed the

following five commands:

```
vi prog.c
cc -c prog.c >err
vi another.c
cc -c another.c >err
link another+prog,prog,prog,c.
lib + /lib/tools.lib
```

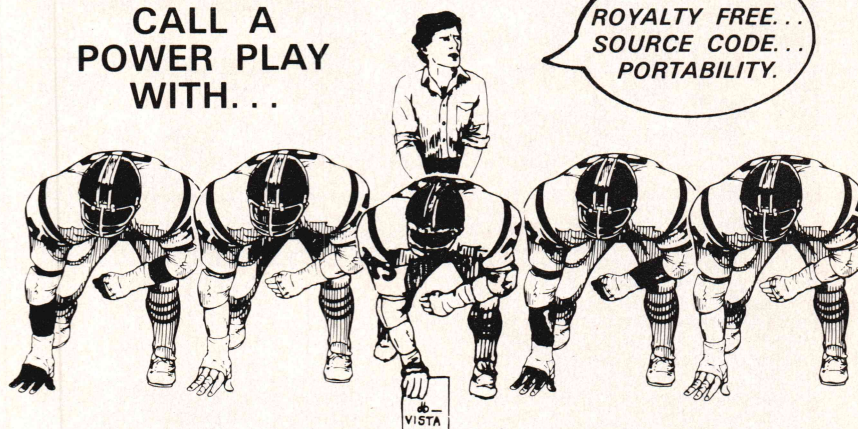
As you type the commands, they're entered into the history list, which can be examined by typing **history**. The following will be printed:

```
1: vi prog.c
```

```
2: cc -c prog.c >err
3: vi another.c
4: cc -c another.c >err
5: link another+prog,,,c.lib + /lib/
  tools.lib
6: history
```

Note that **history** itself is also added to the history list. The numbers are the history numbers associated with each command. A command can be executed again by typing **!#**, where **#** is the history number. Typing **!5** causes the link command to be executed again. **!2** will reexecute the first

C PROGRAMMERS, CALL A POWER PLAY WITH...



...db_VISTA DATABASE MANAGEMENT SYSTEM FOR C.

db_VISTA is a full-featured programmer's DBMS. It handles your data powerfully, yet economically without the frills that make end-user DBMS's bulky, slow and expensive to license. Use only the features you need for maximum efficiency with minimum code and effort.

Powerful lineup of features. B-tree indexing, multiple key records, transaction processing, interactive database access utility, and file transfer utilities for dBASE, R:base and ASCII files. You even get 90 days applications development support free of charge.

Define your playbook up front. As a network model DBMS, **db_VISTA** is suited to applications development. A premium is placed on efficient use of disk storage, reduced data redundancy and fast access times allowing you to cross the goal line first.

It's your game plan. The database structure is specified by you in **db_VISTA**'s data definition language (DDL). The DDL processor compiles the specification into tables (data dictionary) used by **db_VISTA**'s library functions, which are called by your C program to manipulate and access the database.

db_VISTA's written in C so you can understand the signals. Source code is optional. No sweat. No royalties, either.

All this delivered for less than the price of season tickets.

Single user without source\$195
Single user with source\$495
Multi-user without source\$495
Multi-user with source\$990

Available for most popular C compilers under **MS-DOS**, **XENIX**, plus most **UNIX** systems.

Go for the power play and order **db_VISTA** now. Call (206) 747-5570 or

1-800-843-3313

at the tone touch 700-992. 30 day money-back guarantee.

RAIMA
CORPORATION

12201 S.E. Tenth Street
Bellevue, WA 98005 USA
Telex: 9103330300

MACINTOSH VERSION
AVAILABLE SOON

Circle no. 206 on reader service card.

C CHEST:

(Continued from page 19)

cc command. You can also type !<pat>. In this case, the history list will be searched backward for a command starting with <pat>. So !l or !link would also redo the link. lcc would be the same as !4 (because the first matching command is used). !! will repeat the last command you typed. In the above example, !l, !6, and !h will all do the same thing. Commands are added to the history list every time they're executed,

even if a command is an expansion of a history request.

Several additional non-Unix history commands are supported. !>file will write the current history list to the specified file. If no file is given, /histlist is used. The complementary command is !<file, which adds the commands in file to the current history list. The commands aren't executed, they're just added to the list. Again, if file isn't specified, /histlist is used. Neither !> nor !< will show up in the history list (they won't use up a history number,

either).

^ may replace !; that is, the commands ^, ^#, and ^<pat> may be used in place of !, !#, and !<pat>. The ^ commands work just like the ! commands except that the line is brought into an edit buffer that you can then manipulate in the normal way (with the cursors, etc.) before executing it. Unlike DOS, the command line is visible while you're editing it. Note that Esc will abort out of edit mode without executing the edited command line.

Environments, Shell Variables, and the Set Command

Shell variables are macros. They let you associate a body of text with a name, and when that name is used, the corresponding text is substituted. Shell variables are created with the set command and deleted with the unset command. They work something like Unix and DOS environments except that they can't be passed to a child process. Once a shell variable is created, it can be used anywhere in a command. For example, you can define a shell variable to represent a long directory spec with:

```
set HOME /usr/allen/src/shell
```

You can then use it on the command line:

```
cd $HOME
```

Cd \$HOME will be expanded to cd /usr/allen/src/shell before the shell executes the line. Note that \$ must precede all uses of shell variable names but must not be in the definitions; % may be used instead of \$ if you prefer. There are several predefined shell variables (which can't be modified with the set command). These are:

\$<num>—an argument to a batch file (\$0 is the file name, \$1 the first argument, etc.).

\$*—expands to all \$<num> variables concatenated together.

\$p—expands to the current path name.

#!—expands to the current history number.

\$s—expands to the current shell level.

FLASH

HOLIDAY GIFTPACK SPECIAL

NOT COPY PROTECTED

Increase the **speed** of your computer by as much as **4-8** times. Why buy a PC AT when all you may need is some **FLASH**. **FLASH** is just that, it gives disk accesses in a **FLASH**. Every PC owner can't be without **FLASH** once the powers of **FLASH** are discovered. The program is totally invisible to the user. There is nothing to learn. Make your floppies **faster** than a hard disk. Make your hard disks faster than ever before. Allows the user to have sub-directories on his floppies without the usual loss of speed. **FLASH** will also tell you just how much faster you are running than you would have been without **FLASH**. **FLASH** is handcrafted in ASSEMBLER. Upgrade your computer with **FLASH**. Batch files will fly like they had wings. Save wear & tear on expensive program disks. You'll be so **amazed** with your improved performance, you will want to call us to tell us so!

BENCHMARKS for different programs utilizing **FLASH**

	W/O FLASH	W/FLASH
WORD STAR	2.0	0.33
BASICA	2.7	0.40
LINKING	113.8	6.00
SORTING FILES	105.0	27.00

When **FLASH** increases your speed it generally makes you run **3-5** times faster. In favorable conditions like the ones above **FLASH** can make your PC run **6-16** times faster! PC's over the world are screaming for **FLASH**. Reach the speed of a RAM DISK, without their disadvantages. If you think you DON'T need **FLASH**, call us and tell us why!

Order Line 1-800-25-FLASH
Info Line 1-317-253-8088
MC, Visa, Check, C.O.D.
For: IBM PC, AT, XT, JR, or clone.

Reg. Price \$89.95
Limited Time \$49.95
add 4.00 shpg hndl
Software Masters
6223 Carrollton Ave.
Indpls., IN 46220

*** FREE HOLIDAY GIFT ***

When you buy **FLASH**, you get **FREE**, one of the **UTILITY ROOM** programs marked with a ★.

THE UTILITY ROOM

★ Spool-Master , ram printer spooler. Multiple copies, handles single sheets, any buffer size.	49.95	19.95
★ Buffer-Master , keyboard buffer extender from 15 to 2000 key type ahead.	29.95	14.95
★ Super-Ram-Disk , set up MULTIPLE ram disks and as MANY as you want.	49.95	14.95
★ Calc-Pad , ram resident calculator, hex octal binary and decimal. Works in ANY graphic mode. Unlimited parenthesis nesting. Many functions.	49.95	24.95
★ Sort-Merge-Copy , a super sort utility. Use it stand alone or call it from ANY language.	150.00	49.95
★ Unprotect , unprotects basic programs.	49.95	19.95
★ Data-Path , Have paths setup to find your data and overlay files. Works with ALL programs.	49.95	19.95
★ MoreHard , gain up to 10%-50% more hard disk space! ALL programs work in harmony with each other.	49.95	19.95

SUPER—ED Version 3.0

■ Do ANY DOS command or RUN ANY program from within SUPER-ED .	■ Editing commands & colors & more are fully user configurable.
■ Turbo Pascal like environment for ANY language.	■ ON-LINE language-specific user definable HELP .
■ MULTI-FILE window editing. Move & copy between file windows.	■ Full featured editor with WORD STAR like commands.
■ Extensive Find & Replace usage.	■ Fully utilizes sub-directories.
■ ON-LINE editing HELP .	■ Supports keyboard MACROS .

Reg. Price \$59.95
Limited Time \$29.95

We believe in the highest quality software at the lowest possible price!
Turbo Pascal is a trademark of Borland Intl. Word Star is a trademark of Micro Pro Intl.

Circle no. 197 on reader service card.



If you're still paying for multiple copies of dBASE or RunTime[†] at hundreds of dollars a pop, you can stop.

And use a WordTech compiler instead.

With a WordTech compiler, you only need a single copy of dBASE II[†] or dBASE III[†] for every programmer, not every user.

Then you can distribute as many copies of your compiled (and protected) programs as you want with no site licenses, no runtime fees and no strings attached.

Our MS-DOS compilers typically run your dBASE applications 3 to 10 times faster and use just 128K of free memory on any MS-DOS computer, not only the IBM PC.

You don't pay any penalties because WordTech compilers use the same language and syntax as dBASE and the same index, memory and data files (up to 10 data files open simultaneously, each with up to 7 indexes).

Still using dBASE?

And you don't pay the usual high price because WordTech compilers are just \$750. Once.

With a WordTech compiler, you can also run your programs on multi-user and networked systems. Our multi-user compiler runs your existing dBASE programs under AT&T's UNIX System V or CROMIX. And our networking compiler runs your present code on

DOS 3.1 local area networks (LAN's). The compiler will take care of file and record locking for you, or you can take full control using dBASE III networking commands.

We can help you extend your applications as well as your budgets. You can do windows with dBFrame[™],

business graphics with dBChart[™]

and even replace dBASE itself for just \$169 with dB-XL[™], an interpreter with an extended superset of the dBASE III language.

And we back everything with free support, one year of maintenance and a money-back guarantee.

For details and the name of your nearest dealer, contact WordTech Systems, Inc., P.O. Box 1747, Orinda, CA 94563. (415) 254-0900. TELEX 503599.

You owe it to your wallet to call now.

WORDTECH SYSTEMS



C CHEST:

(Continued from page 20)

The top-level shell is 0. If you create a shell within a shell, the second one will be at level 1. All batch files are executed in their own shells.

There are three other shell variables that can be modified with *set* but can't be expanded with *\$name*. These are *echo*, *verbose*, and *cmd*. If *echo* is set (with either a *set echo* or a *set echo = 1*), then commands will be echoed to standard output just before they're executed. The default is *echo*

off (unlike MS DOS), so you must set *echo* inside a batch file if you want to see what the batch file is doing. The echoed line will show all macro substitutions and all wildcard expansions; however, it will be truncated to the 127 characters permitted by DOS. You can use this feature to see if the command line has been truncated when you're invoking a program that doesn't know about the CMDLINE environment (see below).

Verbose shows the input as the shell receives it (before it's interpreted). If *cmd* is set (the default), then an

environment variable called CMDLINE will be created every time a file is executed. CMDLINE holds the entire 2,048-byte command line (which can't be passed via DOS). If *cmd* is cleared, then CMDLINE is still created but will have no contents. *Echo*, *verbose*, and *cmd* can be cleared with *set <name> = 0*. Note that setting *echo* or *verbose* has the same effect as specifying *-x* or *-v* on the command line used to invoke the shell.

Environment variables (or strings) are similar to shell variables except that they can be passed to a child process (a pointer to them is included in a child's PSP). Many compilers (at least the Aztec, Lattice, and Microsoft) have a *getenv(name)* function in their libraries that returns a pointer to the environment string corresponding to *name*. If you need to write your own *getenv()*, a good description of the PSP can be found in *The Peter Norton Programmer's Guide to the Norton IBM PC* (Bellevue, Wash.: Microsoft Press, 1985), pp. 260f.

Environment variables can be set from within the shell with the *setenv* command. Unlike DOS, they can be used on the command line just like a shell variable (precede the name with *\$* or *%*).

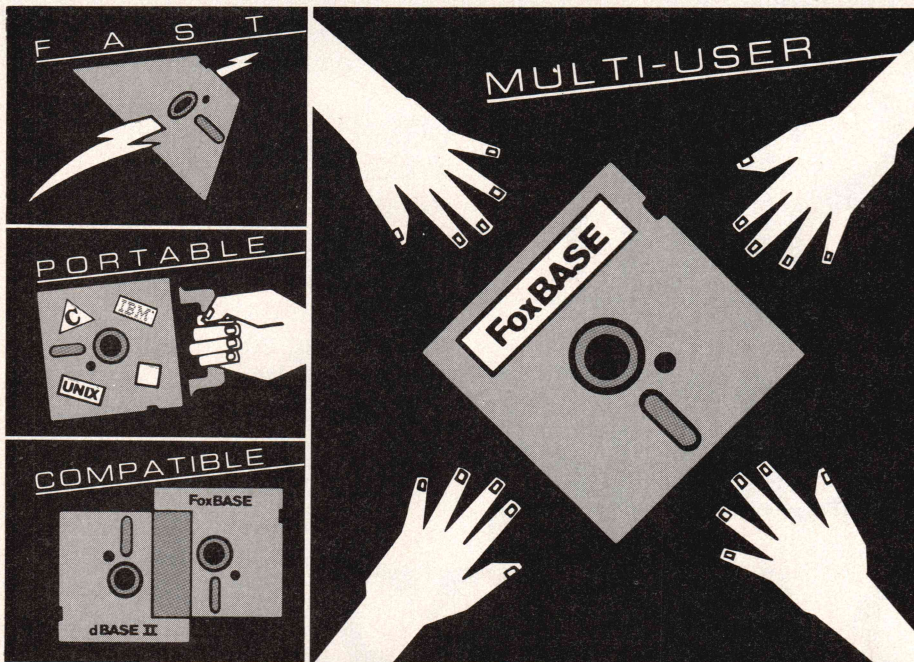
Special Characters

* and ?—have the same significance as in MS DOS. They are expanded by the shell to matching file names. Expanded names are sorted. For example, *echo *.c *.obj* would print a list of all the .c files in the current directory (sorted) followed by all the .obj files (also sorted).

;
—is used to separate multiple commands on a single command line. *Cd foo;pwd* would change the current directory to *foo* and then execute the *pwd* command.

\
—is used to take away the significance of a special character. For example, * can be used to pass an asterisk to *grep* (the * won't be expanded). \; can be used to define a multiple command alias (see below). \\ evaluates to a backslash. The \ will be stripped from the line before the line is passed to the child process.

quotes—text surrounded by double



FoxBASE.

The DBMS That Bridges The Gap Between Single-user And Multi-user.

Unsurpassed Program Development Speed.

FoxBASE™ uses a state-of-the-art B+ Tree index structure for quicker, more efficient data access. A sophisticated virtual storage technique becomes an invaluable timesaving device as it works to insure that frequently referenced programs are retained in memory in compiled form. What's more, FoxBASE provides automatic 8087/80287 math coprocessor support for ultrarapid program execution speed—as much as six times the speed of dBASE II®.

Highly Portable.

FoxBASE is much more than a relational database management system. Written in C, FoxBASE is an extremely portable interpreter/compiler. Now you can port from one machine or operating system to another without changing your applications. And this portability protects your investment in programs by insuring their use in future machine and operating system environments.

dBASE II Compatible.

FoxBASE is both source language—including full macro usage—and data file compatible with the dBASE II database language. This means your existing dBASE II databases can be used unchanged. Furthermore, it puts thousands of public-domain and commercially available dBASE II programs at your disposal.

Available Under Multi-user Systems.

Our multi-user versions of FoxBASE feature many additional enhancements. Like automatic file-locking and record-locking capabilities. Use of termcap, so FoxBASE can run on virtually any terminal. And, with some versions, a two billion record file capacity.

Multi-user Versions:
Xenix® \$995, MultiLink™ \$995.

IBM-PC NET™ \$995.

Single-user Versions:
MS/PC-DOS™ \$395, AOS/VS \$995.

Don't be outfoxed by the others.
Call or write Fox Software today.

FoxBASE™

From
FOX SOFTWARE, INC.

27475 Holiday Lane, Perrysburg, OH 43551
419-874-0162

Circle no. 94 on reader service card.

(') or single (') quotes won't be modified (wildcard characters aren't expanded, semicolons aren't interpreted as command delimiters, etc.). The quotes aren't removed unless the `-q` argument is given on the command line. Unlike Unix, there's no distinction made between single and double quotes.

#—when found in the far left column, signifies a comment. The remainder of the line is ignored, and the line isn't put into the history list.

Aliases

Aliases are another sort of shell-maintained macro. Unlike shell variables, a `$` is not needed to expand the name; rather the alias is expanded if its name is found as either the first word on a line or the first word following a semicolon on a multiple-command line. Aliases have two uses: They can be used to change the name of a command and they can be used in place of batch files. Aliases are created with the `alias` command. Some examples: If you have a program called `ls` that prints the current directory but you also want to be able to type `dir` and get a directory listing, you can define an alias for `dir` as follows:

```
alias dir ls
```

Thereafter, when the shell finds `dir` as the first word on a line, it will substitute the string `ls` for the string `dir`, and the program `ls` will be executed. Only the first word of a command is modified so `dir foo bar rat` will be changed to `ls foo bar rat`. Aliases must be either the first word on a line or the first word following the semicolon when there are several commands on one line.

Aliases can also be used in place of batch files, provided that no arguments need to be expanded. Because aliases are memory resident, they will execute much faster than a batch file. Similarly, an alias doesn't execute under its own shell, as does a batch file, so much less core is needed to execute an alias than is needed to execute a batch file. A simple alias that is similar to a batch file is:

```
alias shell cd /src/util/shell
```

Now you can type the single word `shell`, which the shell will expand to `cd /src/util/shell` and then execute to move to the indicated directory; however, you can't expand arguments. The portion of the command line that follows the alias name will be concatenated to the end of the expanded alias. Let's look at an example: Polymake lets you specify a default rules file on the command line with a `-B` option, but you often have to make a target name as well. By defining the alias `m` with:

```
alias m make -B /lib/  
builtins.mak
```

you can then type `m foo.exe` and the shell will expand it to:

```
make -B /lib/builtins.mak  
foo.exe
```

Because you can have multiple semicolon-delimited commands on one line, you can define an alias that will expand to several commands. For example:

DEBUG FAST!

As a professional, your time is valuable. Advanced Trace86™ is designed as a debugging environment to increase productivity. In many situations, Advanced Trace 86™ can pay for itself in one or two debugging sessions.

Advanced Trace86™ FEATURES

- Trace Screen with user-configurable windows for disassembled code, registers, flags, the stack, the 8087 registers, and memory.
- Single-step tracing with virtually unlimited **back tracing**.
- In-line assembly capabilities, edit .EXE and .COM files in memory (including non-destructive line insertion for .COM files).
- Symbolic debugging capabilities - read or create line labels and comments, read variables.

Debugger Comparison Chart - Selected Features

Features	AT86 Code. Per.	
1. Load Program as memory resident (like Sidekick)	X	X
2. IBM Professional DEBUG & Periscope boards supported	X	X
3. Keyboard break-out support - Ctrl-Enter key combo	X	X
4. Conditional breakpoints	X	X
5. Hex/Decimal calculator & converter	X	X
6. Convert numbers to binary	X	
7. ASCII chart (pop up)	X	
8. In-line Assembler with code insertion capability	X	X
9. 80286/80287 support in Assembler/Disassembler	X	
10. Protect Virtual Mode Support for 80286	X	
11. 8087/80287 window - registers in both decimal & hex	X	
12. Display memory in ASCII, byte, word, & double word	X	X
13. Define "memory structures" for display/editing	X	
14. Try out DOS interrupts command	X	
15. Pop-up Help windows	X	
16. Command macros defined & saved to disk	X	
17. DOS TYPE, DIR, & ERASE commands	X	
18. Search memory for Assembler code - multiple instructions	X	
19. Execute/trace program in reverse & restore machine state	X	
AT86 - Advanced Trace86	Code. - Codesmith	Per. - Periscope

Requires: IBM PC or compatible, minimum 128K RAM. Retail Price - \$175.00

Morgan Computing Co., Inc.

P.O. Box 112730 Carrollton, Texas 75011 (214) 245-4763

C CHEST:

(Continued from page 23)

```
alias m rm err \; make -B /lib/  
      builtins.mak  
  
will expand to  
  
rm err ; make -B /lib/  
      builtins.mak  
  
thereby both removing the file err  
and executing make. Note that you  
must escape the semicolon (with a \)
```

in the alias definition so that the shell won't intercept it. (You could also surround the definition with quotes.)

Aliases may use shell variables. Two such aliases are:

```
alias here set here = \ $p  
alias there cd \ $here
```

Here will set the shell variable *here* to the current directory. *There* will put you in the directory remembered with a previous *here* invocation. Note that *\$* has to be escaped to prevent the shell from expanding it when the

alias is defined.

A caveat about aliases: Aliases defined in terms of other aliases won't work. For example:

```
alias foo echo foo  
alias bar echo bar  
alias foobar foo;bar
```

won't work (*foobar* will expand to *foo; echo bar*). However, the command line *foo;bar* will work. Also note that commands are added to the history list *before* aliases are expanded.

Environments and Files

The default command line prompt is [*\$s,\$!*] (which prints the current shell level followed by the current history number). You can specify a different prompt with the PROMPT environment variable (use either the DOS *set* or the shell's *setenv* command). Any ASCII character may be used, and any of the *\$* arguments will be expanded before the prompt is printed. For example:

```
setenv PROMPT $p->
```

will change the prompt to the current directory name followed by *-* and *>*. The prompt can be changed at any time (it doesn't have to be set when the shell boots). The SWITCHAR environment variable tells the shell what character signifies a command-line switch when it's the first character in a command-line argument (the default is *-*). Because environments are inherited by the child process, SWITCHAR is also available to a program if it chooses to use it.

The CMDLINE environment holds the complete, 2,048-byte command line (which can't be passed via DOS). It is changed every time the shell executes a command. Note that the command line (truncated to 127 characters) is also passed to a child process in the normal way (via the DOS command line buffer at offset 0x80 from the child's initial code segment). When the shell spawns a subshell to execute a batch file, it uses CMDLINE.

The SHLEV environment is set to the current shell level. The outermost shell is at level 0. All shells created from within another shell (including those used to execute batch files) will have higher numbers, de-

Atron's PC/AT Bugbusters

Hardware-assisted Software Debuggers for Bullet-proof PC/AT-based Products

A BUGBUSTER STORY

Brad Crain, a project manager at Software Publishing (the people who developed both PFS:WRITE and PFS:FILE), relates the following: "On Friday, March 22, 1985, I was about to get on an airplane with Jeff Tucker, who was co-author of PFS:WRITE with me, and fly to IBM's Boca Raton, Florida facility. For a week, we had been unsuccessfully trying to isolate a bug in a new software product. In a last, desperation move, I set up an early-Saturday morning appointment with ATRON.

"Three of us walked through ATRON's door at 8:00 the next morning. Using ATRON's hardware-assisted debugging tools, we had the problem identified and fixed by 10:30AM."

Mr. Crain concludes: "We'd never have found the bug with mere

software debuggers, which have the bad habit of getting over-written by the very bugs they're trying to find. It doesn't surprise me that almost all the top-selling software packages were written by ATRON customers. Now that they've broadened their PC family of debuggers to include a PC/AT debugging tool, those of us seriously into 80286 development are greatly relieved."

ARE YOU TRYING TO DO SOMETHING SCAREY?

Like developing your AT-based software product in the dark? Without professional debugging tools?

Seven of the ten top-selling software packages listed by the *THE WALL STREET JOURNAL** were produced by ATRON customers. The PC PROBE™ bugbuster (\$1595) accounts for much of this success. Now that the PC/AT is the new standard for advanced commercial and scientific development, ATRON is proud to announce the AT PROBE™ bugbuster (\$2495). It has even more debugging capabilities than the PC Probe.

HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into your 80287 socket and monitors all processor activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles, you can easily answer the questions: "How did I get here?" and "What are the interrupts doing?"

It can solve spooky debugging problems. Like finding where your program overwrites memory or I/O - impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance-measurement software can isolate your application's bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise demand most of your PC/AT memory.

BORLAND'S PHILIPPE KAHN: "THERE WOULDN'T BE A SIDEKICK™ WITHOUT ATRON'S DEBUGGERS."

So why waste more time reading though your program listing for the ten thousandth time, trying to find why your program starts howling with every full moon. Be like BORLAND, get your Atron bugbuster today and bust bugs tomorrow.



THE DEBUGGER COMPANY

20665 Fourth Street • Saratoga, CA 95070 408/741-5900

*WSJ, June 24, 1985, reporting Softsell figures. © 1985 by ATRON. PC PROBE™ and AT PROBE™ ATRON. SIDEKICK™ Borland. IBM Corp. owns numerous trademarks. Ad by TRBA.

Circle no. 216 on reader service card.

pending on the level of nesting.

Several files are used by the shell. If it exists, the file */shrc.bat* is executed (in a manner analogous to *autoexec.bat*) every time a shell is created. Because batch files are executed in their own shells, */shrc.bat* will be executed every time a batch file is executed. A second file */login.bat* is executed only once, when the lowest-level (level 0) interactive shell is created. *Shrc.bat* is executed before *login.bat*, and both files are executed before the environment is examined. My own *login* file is shown in Table 1, at right. My *logout* file is simply `!>` and lets me leave the shell without losing the current history list. Note that `!<` in *login.bat* lets me reenter the shell without losing the list. You could also use `!<` (without the `!>`) to read in a list of commonly used commands when the shell boots.

Shell Invocation Syntax

There are several ways to get into the shell from the command line. The easiest way is to type *sh* (with no arguments), putting you into *interactive* mode. You can't get out of the lowest-level interactive shell with a `^C`. Use either *exit* or *logout*.

sh -c string—invokes the shell in nonresident mode. It will execute the command contained in *<string>* as if it had been entered from the command line and then terminate.

sh filename args . . .—executes a batch file. *\$0*, if found inside the batch file, will be expanded to the *file name*. The arguments can be fetched with *\$1*, *\$2*, etc. *%* can be used instead of *\$* if you like.

Four other command-line switches are available:

-i—puts the shell into interactive mode even if arguments are listed on the command line. Normally, if command-line arguments are present and *-c* isn't specified, the shell will try to execute a batch file. *sh -i arg . . . arg* will create an interactive shell, putting the arguments into *\$1*, *\$2*, etc. *\$0* will hold the string *-i*.

-q—causes quotes to be stripped from commands before they're passed to a child process. The

```
setenv PROMPT=[\ $s:\$!]  
alias a alias  
a h history  
a book      cd /text/book  
a ddj       cd /text/ddj  
a here      set here = \$p  
a sclass    cd /src/class  
a there     cd \$here  
a tmac      cd /lib/tmac  
a tools     cd /src/tools  
a type      cat  
a m         "rm err; make -B builtins.mak"  
!<
```

Table 1: A *login.bat* file

- quotes will still protect wildcard characters, etc., from expansion.
- v*—verbose mode, commands are echoed to *stderr* as they're read by the shell. This is the same as a *set verbose* command.
 - x*—commands are echoed to *stderr* just before they're executed. All *\$* arguments and wildcard characters will have been expanded at this point. This is the same as a *set echo* command.

Redirection

The shell itself doesn't support redirection; however, because *command.com* is still resident, redirection is available if you need it. There are two ways to use *command.com* for redirection. A nonresident shell can be invoked from DOS with a line such as:

```
sh -c grep pattern *.c >foo
```

Grep will be executed, and the shell will expand the **.c* to the names of all files in the current directory having a *.c* extension before *grep* is invoked. *Grep*'s output will be redirected to *foo* in the normal way. Because this command is executed from MS DOS and not from the shell, it won't be added to the history list.

The second method also lets you enter redirected commands into the history list. From inside the shell, type:

```
command /c grep pattern *.c >foo
```

The */c* argument to *command.com* works like the *-c* argument to *sh*, so it will execute the following string as if it had been typed. The **.c* will again be expanded by the shell before *command.com* is executed, but the *>* will be interpreted by *com-*

mand.com (which will put the output into *foo*). Be careful here of command-line truncation. Because *command.com* doesn't know about the CMDLINE environment, it has no way to get to the extended command line, so it will work on only the first 127 characters.

Support Routines

In order to minimize the size of the shell, I've tried not to build in commands that aren't essential. I've found the following programs to be useful.

- cat.c*—prints (concatenates) files to *stdout*.
- cp.c*—copies a file to another file or disk. Copies a group of files to another directory or disk.
- echo.c*—echoes command line to *stdout*.
- grep.c*—searches for pattern in file.
- ls.c*—lists a directory.
- mkdir.c*—creates a directory.
- mv.c*—renames a file or moves a group of files to another directory.
- printenv.c*—prints the current environment.
- rm.c*—removes a file or group of files.
- rmdir.c*—removes a directory.

Availability

This column is part of a four-part series describing the entire shell. A reprint of all four parts along with a disk containing the listings is available for \$29.95 from *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. Please direct inquiries to The Shell. Prepayment is required.

DDJ

(Listing begins on page 84)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 1.**

PL/68K C Becomes 68000 Assembly Language

by Edward K. Ream

One day not long ago, I became embroiled in an old debate with another programmer named Charlie. . . .

"The programming team I manage is about to start a big project," I said, "and I must decide which language to use."

"Really? Which languages are you considering?"

"C and 68000 assembly language. The product will have strong competition, and great performance is crucial, so it's reasonable to consider assembly language. On the other hand, C is so much easier to use."

"Why don't you program in C and recode in assembly language as needed?" Charlie asked.

"Of course I've considered that. It might work as far as execution speed is concerned, although I'm not sure. C doesn't let you allocate registers globally, and that's a big handicap. Speed is not the only problem, though. The code must be compact, but our C compiler produces code that is 50 percent larger than assembly language. No, there's no doubt about it—eventually the program will have to be written in assembly."

I asked myself, suppose the program produced by the C compiler and the program produced by the assembler were semantically equivalent? Suddenly PL/68K became not just another assembly language but a new way of using C.

"Do the initial prototyping in C. That's the right way," Charlie persisted. "When the program is finished, recoding in assembly language will be much easier."

"Hmmm. I'm not convinced. Recoding is going to be expensive; we'll end up debugging the whole program twice. There

might even be pressure from higher management not to recode and come out with an inferior product."

Charlie just snorted and walked away, muttering something about assembly language being a throwback to the Dark Ages.

Writing in Both C and Assembly

Fortunately, my friend John overheard this conversation. John and I have worked together for 15 years, and we enjoy discussing problems that come up on the job. John laughed, "Charlie is more interested in being right about C than in solving your problem."

"You sound more sympathetic."

"Well, your choice is crucial. Which language you use determines, to a large extent, how your project will turn out."

"Yes. What bothers me most is that I've got to choose now, but I won't know until the project is almost over whether the choice was correct."

"I think I know a way around this dilemma—it's a language I invented called PL/68K."

"John, my only options are C and 68000 assembly language."

"Don't be fooled by the name. PL/68K isn't really an independent language but a way of using C to do assembly-language programming."

"John, you are not making sense!"

"Let me explain. You can think of PL/68K as being either C or assembly language—either/or. But in fact, you can run a program written in PL/68K through both the PL/68K assembler and any standard C compiler. PL/68K is both C and assembly language at the same time."

"Wait a minute. You are going too fast," I said. "First of all, you can't possibly compile an assembly-language program with a C compiler! Assembly language doesn't look anything like C—the C compiler will spit out a thousand error messages!"

"PL/68K doesn't resemble 'traditional' assembly language. Forget what assembly language usually looks like and ask yourself, 'What are the characteristics of assembly language?'"

"Go on," I replied. "You tell me."

"First, assembly language allows full access to all machine resources—all registers, all locations in memory (including the run-time stack), all I/O ports, all privilege modes, and all machine instructions. Second, there is a one-for-one correspondence between the source code you write and the object code produced by the assembler. You always know what code a particular assembly-language construct generates; assemblers neither rearrange code nor 'optimize' code away nor add anything extraneous. Assemblers are very literal-minded. Thus, assembly language ensures zero time and space overhead."

"You're saying that assembly language gives you complete control over the machine, without a compiler getting in the way."

"Exactly. Now, suppose we say assembly language is any language that (1) allows complete access to all machine resources, (2) provides a clear correspondence between source code and object code, and (3) imposes zero time or space overhead."

Semantic Identity

"Hmmm," I mused. "This definition doesn't say what assembly language looks like. It could even look like C. But I still don't understand. If you run a PL/68K program through an assembler, you will get one program. If you run the same source through a C compiler, you will get a second program. The two programs are not going to do the same things—similar things, maybe, but not the same things. The fact that the source code is the same doesn't matter. To put it another way, given a result desired from a specific PL/68K program, we would still have to choose between assembling the program with the PL/68K assembler or compiling it with a C compiler."

"You've stated the problem very well," John said, "but I have discovered that it's possible to design PL/68K so that the program produced by the PL/68K assembler will work in the same way as the program produced by the C compiler."

"That sounds impossible!"

"I don't think so. Let's turn the problem around. Suppose we design PL/68K according to what might be called 'the principle of semantic equivalence.' This principle states that a program, when assembled by the PL/68K assembler, must work in the same way as when it is compiled by a standard C compiler. Now let's ask, 'What needs to be eliminated from PL/68K to guarantee semantic equivalence?'" (See Figure 1, page 40.)

"Tell me," I said, "how much of C is left after the principle of semantic equivalence takes its toll?"

"Surprisingly, almost all of it. The preprocessor is identical to the C preprocessor. All declarations and structure statements are present. Functions do not return values but otherwise are unchanged, as are Boolean and relational operators and expressions. The biggest restriction is that arithmetic operators and expressions must be severely curtailed in order to make PL/68K expressions mean the same thing as C expressions."

"You keep talking about PL/68K being assembly language," I said. "How is it possible to produce code the quality of assembly language from a language that is a subset of C?"

"I haven't shown you the whole language yet. Two other rules guide the design of PL/68K. These rules, together with the principle of semantic equivalence, determine the form and content of PL/68K. The two rules are 'the code selection rule'—the assembler for PL/68K does no code selection, and all arithmetic operations in PL/68K correspond to unique 68000 machine instructions; and 'the register allocation rule'—the assembler for PL/68K does no register allocation, and all operations in an assignment statement are performed in the location specified by the left side of that assignment statement."

"In short, these rules say that an assembler for PL/68K never has to make any significant decisions. Because the PL/68K assembler knows how to select code and allocate registers, it will never need any of the fancy techniques used by optimizing compilers, but it will be able to produce code that is just as good as assembly language."

"I like to think of the assembler for PL/68K as a simple compiler, consisting of a parser and straightforward code generator and possibly a peephole optimizer. The whole job should take about a year to complete rather than the 10 to 20 programmer years for a typical optimizing compiler. Using compiler technology to write an assembler was the initial idea that started me thinking about PL/68K."

Now that I've presented the general ideas behind PL/68K, I'll drop this dialogue format and these fictional characters and look at the details of the language.

Specifying Registers

Because PL/68K is both assembly language and C, some way must be found to deal with assembly-language constructs such as registers, address modes, and individual machine instructions, while at the same time remaining compatible with C. These assembly-language constructs are represented by reserved words, shown in Table 1, page 29.

As for the registers of the 68000, d0 through d7 stand for the data registers, a0 through a7 for the address registers, pc for the program counter, ssr for the status regis-

M68000 SINGLE BOARD COMPUTER



On board 6-10 MHz CPU, 20K RAM, 32K EPROM, two RS-232, 16-bit port, 5-counter/timers expandable via Memory/FDC Board.

M68K CPU (bare board)	\$ 89.95
M68K CPU A&T (6MHz)	\$495.00
MD512K Memory/FDC (bare board)	\$ 89.95
MD512K Memory/FDC (128K)	\$495.00
FDC/Hard Disk interface option	\$150.00
M68KE Enclosure w/power supply	\$249.00
M68K Monitor EPROM's	\$ 95.00
M68K Macro Cross Assembler	\$195.00
4XFORTH OS w/assembler, editor	\$295.00
CP/M 68K OS w/'C' compiler	\$395.00

EMS

Educational
Microcomputer
Systems

P.O. Box 16115
Irvine, CA 92713
(714) 854-8545

Circle no. 210 on reader service card.

Now! Automatic time and
date stamping for
CP/M® 2.2

DATESTAMPER™

- Extends CP/M 2.2 to automatically record date and time a file is created, read and modified. DateStamper reads the exact time from a real-time clock if you have one, or records the order in which you use files each day.
- Datestamping information is contained in a separate file and read by our directory utility, SDD.COM. Powerful DATSWEEP file management utility also included.
- Simple menu-driven installation. Large library of clocks supplied, with provision to add others. Many options configurable to your individual requirements.
- Disks prepared for datestamping are fully compatible with standard CP/M. DateStamper also runs with all versions of ZCPR, Z-RDOS and many other CP/M-80 modifications.

CP/M is a registered trademark of Digital Research, Inc.

Plu*Perfect Systems

8" SSSD, Kaypro, Osborne, H/Z-89 formats \$49
(Most other formats, add \$5)
Shipping & handling \$3
California residents add 6% sales tax
MasterCard & Visa accepted

Avoid
erasing the
wrong files!

Back up
files by time
and date!

"DateStamper...
is a real winner."

Bruce Morgen, Users
Guide, Jul-Aug. 1985

Write or call for
further information

(714) 659-4432

BOX 1494, IDYLLWILD, CA 92349

PL/68K

(Continued from page 27)

ter, and ccr for the condition code register, which is the lower byte of the ssr.

Standard aliases are also defined. Register a7 can also be called sp, ssp, or usp to denote the stack pointer (or system stack pointer or user stack pointer). The reserved words r0 through r7 are synonyms for d0 through d7, and the names r8 through r15 are synonyms for registers a0 through a7.

All registers on the 68000 are 32 bits long (except the status registers), but not every instruction uses all 32 bits of a register. Besides long (32-bit) operations, byte-length (8-bit) and word-length (16-bit) operations are permitted on data registers, and word-length operations are permitted on address registers. To represent the length of an operation, the name of any data register can be followed by a *b* to denote byte length or a *w* to denote word length. Thus, *d0* stands for the long register d0, *d0w* stands for the word-length register d0, and *d0b* stands for the byte-length register d0. Address registers are treated in a similar manner, except that byte-length operations are not permitted.

Address Modes

The 68000 has 12 different address modes, or means of accessing operands. (See Table 2, page 29.) The address modes are represented in PL/68K by five of C's operators, namely &, *, ++, --, and -->.

Let's look, for example, at the Address Register Indirect with Postincrement address mode. (It's a lot easier to use than to say.) This mode uses the contents of an address register as the address of an operand. After the operation is performed, the address register is incremented by 1, 2, or 4, depending on the size of the operation. In traditional assembly language, that mode applied to address register a0 would be written as (a0)+. In PL/68K, that address mode is represented by *a0++. For example, you would write *d0b=*a0++*; in PL/68K instead of *move.b (a0)+,d0b*. Constructions such as *++a0 are not allowed because of the code selection rule. The 68000 has no addressing mode of the form +(a0), so *++a0 is not part of PL/68K.

The word *primitive* denotes what is called an effective address in machine-language terms. A primitive describes an operand, which may be in a register, on the run-time stack, or in static memory. In PL/68K, the valid forms of primitives are determined by the address modes I've just discussed.

Declarations

While I am talking about operands, I'll say a few words about how those operands are declared. Declarations in PL/68K are just the same as in C, except that functions do not have types. If you think about it for a moment, this means that PL/68K declarations have no parentheses. Declarations can never become unreadable as they can in full C.

In effect, declarations produce DC (define constant) and DS (define storage) pseudo-operations. (See Table 3, page 29.) Although declarations produce no executable code,

they determine what code gets produced by arithmetic operators. For instance, if *a* is an integer, the assignment statement *a* *= *b*, which multiplies *a* by *b*, generates a MULS (signed multiply) machine instruction, but if *a* is an unsigned integer or pointer, the assignment statement generates a MULU (unsigned multiply) instruction.

As another example, the assignment *a* += *b*, which adds *b* to *a*, generates an ADD.B (byte length add) instruction if *a* is a *char*, but it generates an ADD.W (word length add) instruction if *a* is an *int* and generates an ADD.L (word length add) instruction if *a* is a long word or a pointer.

Assembly-Language Instructions and Pseudo-operations

Reserved identifiers also stand for 68000 machine-language instructions and pseudo-operations. A library of pseudofunctions must be linked with a PL/68K program when it is translated with a C compiler. This library, called the ops library, contains declarations and functions that allow C programs to simulate the effect of 68000 machine instructions and pseudo-operations. (See Table 4, page 32.)

The pseudofunction *btst*(), for example, simulates the BTST (bit test) machine instruction. In PL/68K, you would write *btst(1,d0b)*; in those places where you would write *btst.b #1,d0* in traditional 68000 assembly language.

Other pseudofunctions allow PL/68K programs to refer to assembly-language pseudo-operations. The PL/68K assembler translates the *org*(), *even*(), *bss*(), *text*(), and *data*() pseudofunctions to the ORG, EVEN, BSS, TEXT, and DATA pseudo-operations. Similarly, the PL/68K assembler translates the *dcbl*(), *dcw*(), *dcl*(), *dsb*(), *dsw*(), and *dsl*() pseudofunctions to the DC.B, DC.W, DC.L, DS.B, DS.W, and DS.L pseudo-operations. None of these pseudofunctions has any effect when a C compiler translates a PL/68K program. In other words, the corresponding pseudofunctions in the ops library do nothing.

You may be wondering why I keep calling these routines pseudofunctions. After all, they are perfectly good functions when compiling a PL/68K program with a C compiler. When you turn the program through the PL/68K assembler, though, it translates pseudofunctions directly into 68000 machine instructions or pseudo-operations.

Expressions and Assignment Statements

I've covered the components of low-level assembly language—registers, address modes and effective addresses, machine instructions, and pseudo-ops. Now let's see how you put these components together to make expressions and assignment statements.

Expressions are quite restricted; they are just primitives or parenthesized constant expressions. Assignments are restricted to the forms *primitive aop expression* or *primitive aop (assignment)*, where *aop* stands for one of the assignment operators of the C language, namely, =, +=, -=, *=, /=, &=, |=, ^=, >>=, and <<=. The regular arithmetic operators in C, namely, +, -, *, /, &, ^, ^, >>, and << are allowed only in constant expressions, which must be parenthesized.

You are probably wondering why all these restrictions exist. There's a short answer and a long answer. The

Data registers

d0, d0b, d0w, . . . , d7, d7b, d7w
r0, r0b, r0w, . . . , r7, r7b, r7w

Address registers

a0, a0w, . . . , a7, a7w
r8, r8w, . . . , r15, r15w
sp, spw, usp, uspw, ssp, sspw

Status registers

ssr, ccr

Program counter

pc

Table 1: Reserved words corresponding to 68000 registers

PL/68K	Traditional assembly language
123	#123
abc	abc
&abc	#abc
*(&abc+1)	abc+ #1
abc.25	abc+ #25
*(0x80)	\$80
a0	a0.l
a0w	a0.w
d0	d0.l
d0w	d0.w
d0b	d0.b
*a0	(a0)
*a0++	(a0)+
*--a0	-(a0)
a0 → 5	#5(a0)
a0 → d0	#0(a0, d0.l)
a0 → (d0w)	#0(a0, d0.w)
a0 → (d0+5)	#5(a0, d0.l)
a0 → (d0w+5)	#5(a0, d0.w)
pc → 5	#5(pc)
pc → d0	#0(pc, d0.l)
pc → (d0w)	#0(pc, d0.w)
pc → (d0+5)	#5(pc, d0.l)
pc → (d0w+5)	#0(pc, d0.w)

Table 2: Representing the address modes of the 68000

Declaration	Code Generated
char abc;	abc: ds.b 1;
char c1 = 'c';	c1: dc.b 'c';
char *cp;	cp: ds.l 1;
long xyz;	xyz: ds.l 1;
char a[] = "abc";	a: dc.b 'abc',0;
int a2[25];	a2: ds.w 25;
struct s1 { long sl; char *s2; int s3; };	no code generated.
struct s1 s2[25];	s2: ds.b 250;
union u1 { int ui; char uc; long ul; };	u1: ds.b 4;

Table 3: Code generated for declarations

short answer is that it's the only way to reconcile all of the design rules behind PL/68K with all of C's rules concerning operator precedence and arithmetic conversions. Take my word for it: You cannot have full C expressions in PL/68K and still retain the principle of semantic equivalence. For the long answer to this question, see the Appendix section.

There are additional restrictions concerning assignments to pointers. Suppose, for example, that *a0* has been declared to be a pointer to some object *X* whose size is larger than 1. In this case, the only assignments allowed to *a0* are assignments of the form

a0 += constant;

(or constant expression) or

a0 -= constant;

To conform to C's scaling rules, these constructions are equivalent to

ADDA CONSTANT * SIZEOF(X), A0

SUBA CONSTANT * SIZEOF(X), A0

If you do not want this scaling to take place, you can use the *adda()* or *suba()* pseudofunctions. In general, no scaling, optimization, or other kind of tampering is ever done to the arguments of pseudofunctions. This rule has no exceptions.

Two special forms of assignment statements are permitted: *location++*; and *location--*;, where *location* stands for an address register, data register, or the name of a memory variable.

A few words about why PL/68K does not have function calls and array references—let's consider function calls first. Functions don't return values in a designated location because no single spot is right in all cases. Returning the results of a function in a register would conflict with the register allocation rule. Thus, the programmer must specify how functions will return values. Note, however, that you can pass arguments to a function in the normal way, as you shall see shortly.

The array operator *[]* conflicts with both the register allocation and the code selection rules. There is no suitable spot in which to evaluate subscript expressions, and picking a spot at random would violate the register allocation rule. (Actually, no single location would suffice because general C expressions can require arbitrarily many temporary locations to evaluate.) In addition, the "natural way" to access arrays on the 68000 is with pointers, which is how an assembly-language programmer would probably do it. Thus, retaining the array operator would violate the spirit behind the code selection rule.

Assignment statements are simple, but that shouldn't hurt much—assignment statements usually have only one or two operators anyway.

As the code selection rule requires, PL/68K defines what code arithmetic expressions will generate. (See Ta-

ble 5, page 32.) The += operator generates the ADD instruction, the -= operator generates the SUB instruction, and so on. Many machine instructions on the 68000 have several variants—for example, ADDA adds address registers, ADDI adds literal data, and plain ADD adds to data registers and memory locations. The assembler has to do some code selection, but the choice is easy; even traditional assemblers do that much.

Boolean Expressions

Boolean expressions in PL/68K are similar to Boolean expressions in C. All the Boolean operators *!*, *||*, and *&&* and all the relational operators *=*, *!=*, *<*, *<=*, *>*, and *>=* are allowed. Of course, arithmetic expressions are restricted in Boolean expressions just as they are in assignment statements.

Boolean expressions in PL/68K are not general expressions, as they are in C. Boolean expressions can only appear in the appropriate part of *if*, *do*, *while*, and *for* statements. The code fragment *a == b*; is not valid in PL/68K outside a structure statement. This restriction eliminates a whole class of hard-to-find errors (the programmer almost certainly meant to say *a = b*).

Because Boolean expressions appear in limited contexts, much less work is required to evaluate them. (See Tables 6 and 7, page 32.) Boolean expressions of the form *primitive* or *primitive relop 0* or *0 relop primitive* (where *relop* denotes one of the relational operators) generate the TST (test operand) instruction followed by some form of the *Bxx* (conditional branch) instruction. The *Bxx* instruction chosen depends on the *relop* and the declared type of the operand being tested.

Expressions of the form *primitive relop nonzero constant* or *nonzero constant relop primitive* generate the CMP (compare) instruction followed by a *Bxx* instruction.

The NOT operator generates no code at all but instead simply reverses the "polarity" of one or more *Bxx* instructions. For example, the statement

if (*a == 0*)

generates

TST A
BNE

while the statement

if (!(*a == 0*))

generates

TST A
BEQ

Similarly, the *||* and *&&* operators generate no extra code. Incidentally, you can use parentheses in Boolean expressions to affect the order of binding of Boolean operators. For instance,

if (*a && !(b < 5 || b > 20)*)

A FULL C COMPILER FOR \$49.95

Limited time offer:
**FREE CED
TEXT EDITOR
WITH EACH ECO-C88**



Ecosoft's Eco-C88 C Compiler

An unbeatable value! For \$49.95 you set:

- A C compiler with all data types and operators (except bit fields)
- Fast executing code. Some common benchmark results are:

	Eco-C88	L (1)	C86 (1)	MS (1)	MW (1)
sieve	12	11	13	11	12
fib	43	58	46	109	—
deref	14	13	—	10	11
matrix	22	29	27	28	29

1. Computer Language, Feb., 1985. Reproduced with permission.

- 8087 support using a single library. The 8087 is sensed at runtime and used if present.
- A standard library with over 200 functions (many of which are System V compatible for greater portability).
- Error messages in English - no cryptic numbers to look up.
- A cc and "mini-make" (in source) that makes compiling a snap.
- ASM or OBJ output (for use with MSDOS linker).
- Expanded user's manual.
- Works with all IBM PC's and clones using MSDOS 2.1 or later.

ORDERS ONLY

1-800-952-0472

The ECOSOFT family of C products ORDER FORM

- ☐ C Compiler \$49.95 _____
- ☐ Programming Guide \$20.00 _____
- ☐ Self-Study Guide \$17.00 _____
- ☐ Programmer's Library \$20.00 _____
- ☐ Program Editor \$29.95 _____
- ☐ C Library Source \$10.00 _____
- ☐ ISAM (.OBJ) \$15.00 _____

Total* (Ind. res. add 5% tax) _____

*Please add \$4.00 for shipping.

Payment: VISA MC AE Check

Credit card expir. date _____

Card # _____

Name _____

Address _____

City, state _____

Zip _____ Phone _____

Ecosoft, Inc.

6413 N. College Ave.
Indianapolis, IN 46220

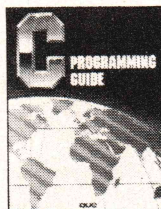
(317) 255-6476 • 8:30-4:30



Other Eco-C Products

CED Program Editor. \$29.95

A screen-oriented program text editor similar to the Turbo Pascal editor. You can create, compile and link the source file with CED. If there is an error, CED automatically reloads the source file and places the cursor on the offending section of code. CED supports editing multiple files (with windows), macros, more than 50 editing commands and is configurable to your particular needs and preferences. An outstanding value.

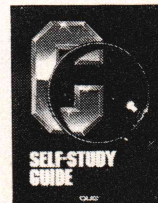


C Programming Guide, 2nd Ed. (Purdum, Que Corp.) \$19.95

This best seller walks you through the C language in an easy-to-read manner. All aspects of the language are covered, including many of the new ANSI Standards suggestions. Many of the error messages issued by the Eco-C88 compiler reference page numbers in this text making an ideal learning environment.

C Self-Study Guide (Purdum, Que Corp.) \$16.95

This new book is designed for the person that is learning C on their own. The book is filled with questions and answers that most beginning C programmers have. It also includes many sample programs that illustrate tips, traps and techniques that may take years to discover otherwise. A perfect compliment to the Guide book.



C Programmer's Library

(Purdum, Leslie, Stegemoller, Que Corp.) \$19.95

This best seller is an intermediate level text designed to show you how to write functions for your C library. The book contains many useful library additions, including an ISAM file handler, plus sections on advanced C topics.

TRADEMARKS: ECO-C88, ECOSOFT, TURBO PASCAL, BORLAND INT'L.

Pseudofunctions corresponding to 68000 machine instructions

abcd, add, adda, addi, addq, addx, and, andi, asl, asr
 bcc, bcs, bchg, beq, bge, bgt, bhi, ble, bls, blt, bmi, bne, bpl,
 bvc, bvs
 bchg, bclr, bra, bset, bsr, btst
 chk, clr, cmp, cmpa, cmpi, cmpm,
 dbcc, dbcs, dbeq, dbf, dbge, dbgt, dbhi, dble, dbls, dblt, dbmi,
 dbne, dbpl, dbt, dbvc, dbvs, divs, divu
 eor, eori, exg, ext, jmp, jsr, lea, link, lsl, lsr
 move, movea, movem, movep, moveq, muls, mulu
 nbcd, neg, negx, nop, not, or, ori, pea
 reset, rol, ror, roxl, roxr, rte, rtr, rts
 scc, scs, seq, sf, sge, sgt, shi, sle, sls, slt, smi, sne, spi, st, svc,
 svs
 sbcd, stop, sub, suba, subi, subq, subx, swap
 tas, trap, trapv, tst, unlk

Pseudofunctions corresponding to assembly-language pseudo operations

dcb, dcw, dcl, dsb, dsw, dsl
 org, data, text, bss, even

Table 4: Pseudofunctions

Operator	Generated code
a += b	add b, a (or adda or addi or addq)
a -= b	sub b, a (or suba or subi or subq)
a * = b	muls b, a (or mulu)
a / = b	divs b, a (or divu)
a % = b	divs b, a (or divu)
	swap a
a >> = b	asr b, a (or ror)
a << = b	asl b, a (or rol)
a & = b	and b, a (or andi)
a = b	or b, a (or ori)
a ^= b	eor b, a (or eori)
a + +	addq #1, a (or addi)
a - -	subq #1, a (or subi)

Table 5: Code generated by arithmetic operators

Boolean	Generated code
if (Z)	bnz false
if (a)	tst a (or cmpa) bz false
if (a < b)	cmp b,a bge false
if (!a)	tst a bnz false
if (a && b)	tst a beq false tst b beq false
if (!(a && b))	tst a beq true tst b bne false true:

Table 6: Code generated by Boolean expressions

Signed Comparisons

Unsigned Comparisons

if (c1 == c2)	
cmp c2,c1	cmp c2,c1
bne false	bne false
if (c1 != c2)	
cmp c2,c1	cmp c2,c1
beq false	beq false
if (c1 < c2)	
cmp c2,c1	cmp c2,c1
bge false	bcc false
if (c1 < = c2)	
cmp c2,c1	cmp c2,c1
bgt false	bhi false
if (c1 > c2)	
cmp c2,c1	cmp c2,c1
ble false	bls false
if (c1 > = c2)	
cmp c2,c1	cmp c2,c1
blt false	blo false

Table 7: Code generated by Boolean comparisons

Macro Name	Pseudo-function	Meaning
Z or EQ	cc_z	zero
NZ or NE	cc_nz	not zero
C or CS	cc_c	carry
NC or CC	cc_nc	no carry
V or VS	cc_v	overflow
NV or VC	cc_nv	no overflow
GT	cc_gt	greater than
GE	cc_ge	greater or equal
LS	cc_ls	less than
LE	cc_le	less than or equal
HI	cc_hi	high
LO	cc_lo	low
MI	cc_mi	minus
PL	cc_pl	plus

Table 8: Condition code constants

Syntax

(1) if (boolean) { statement list }
 (2) if (boolean) { statement list 1 } else { statement list 2 }

Code generated for (1)

\$ Evaluate boolean. If false, jump to label 1 \$
 \$ statement list \$
 label1:

Code generated for (2)

\$ Evaluate boolean. If false, jump to label 1 \$
 \$ statement list 1 \$
 bra label2;
 label1:
 \$ statement list 2 \$
 label2:

Table 9: The if statement

PL/68K
(Continued from page 30)

is valid and generates

```
TST A
BEQ
CMPI 5, B
BLT
CMPI 20, B
BGT
```

You can specify condition code values directly. (See Table 8, page 32.) For instance, the statement *if (Z)* tests the current value of the zero bit in the condition code register and generates the BNZ (branch not zero) instruction. *Z* is a macro in C, defined in the ops library, which expands to a call to the pseudofunction *cc_z()*.

Structure Statements

I said earlier that PL/68K has all C's structure statements—*if*, *do*, *while*, *for*, and *switch*. They look exactly like C code, but curly braces are required surrounding statement lists in structure statements. In other words, structure statements have the form

```
if( . . ) {statement list}
if( . . ) {statement list} else {statement list}
while( . . ) {statement list}
do {statement list} while( . . );
for( . . ) {statement list}
switch( . . ) {statement list}
```

In my opinion, allowing curly braces to be optional is a big flaw in C. In this example:

```
if (abc < 5)
    xyz = 5;
if (abc < 6)
    xyz = 6;
```

the indentation is misleading and will probably cause a bug. This kind of error can be extremely difficult to find.

Let's see what code PL/68K's structure statements produce. In the accompanying tables, the dollar sign denotes code that corresponds to some language construct. For example, *\$ statement list \$* stands for whatever code is generated for the statement list. The statement list could be arbitrarily complicated—for instance, it could contain nested structure statements. The notation

\$ Evaluate boolean. If false, jump to label1 \$

indicates that code is generated for the Boolean expression such that a jump to *label1* is taken if the Boolean expression is false. Otherwise, control falls through to the following code. Labels are indicated in the usual way, by identifiers followed by colons. All generated labels are, of course, unique, even though they may have identical names in the tables.

Table 9, page 32, shows the *if* statement. When an *if* statement contains no *else* clause, the Boolean expression

Make your Mac run like a Lisp Machine with MacScheme™

HIGH QUALITY BYTE CODE INTERPRETER

Fast as interpreted Lisp on Lisp machines
Compact code for large programs
Reliable error reporting and recovery
Debugger, trace facility
Escape to machine code for full access to Toolbox

EASY TO USE

Edit and evaluate expressions in any window
Editor matches parentheses, indents automatically
Needs only one drive, 512K or more RAM

CONFORMS TO 1985 SCHEME STANDARD

Lexically scoped with block structure
True first class procedures
Generic arithmetic: fixnums, flonums, bignums
Plus strings, vectors, arrays, ports, and more

GREAT PRICE: \$125 (Not copy protected)

Semantic Microsystems



4470 S.W. Hall St., Suite 340
Beaverton, OR 97005
(503) 643-4539

Circle no. 217 on reader service card.

¿C? ¡SÍ!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can use on **any** machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only \$18 for a full year (regularly \$28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal**!

Please add \$9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal**: InfoPro Systems. UNIX: AT&T Bell Labs.



InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570



Circle no. 194 on reader service card.

Syntax

- (1) while (*boolean*) { *statement list* }
- (2) while (1) { *statement list* }

Code generated for (1)

```
bra continue_label;
label1:
$ statement list $
continue_label:
$ Evaluate boolean. If true, jump to label 1 $
break_label:
```

Code generated for (2)

```
continue_label:
$ statement list $
bra continue_label;
break_label:
```

Table 10: The while statement

Syntax

- (1) do { *statement list* } while (*boolean*);
- (2) do { *statement list* } while(1);

Code generated for (1)

```
label1:
$ statement list $
continue_label:
$ Evaluate boolean. If true, jump to label 1 $
break_label:
```

Code generated for (2)

```
continue_label:
$ statement list $
bra continue_label;
break_label:
```

Table 11: The do statement

Syntax

- (1) for (*assignment list 1*; *boolean*; *assignment list 2*) { *statement list* }
- (2) for (*assignment list 1* ; ; *assignment list 2*) { *statement list* }
- (3) for (*assignment list 1* ; 1; *assignment list 2*) { *statement list* }

Code generated for (1)

```
$ assignment list 1 $
bra label0;
label1:
$ statement list $
continue_label:
$ assignment list 2 $
label0:
$ Evaluate boolean. If true, jump to label 1 $
break_label:
```

Code generated for (2) or (3)

```
$ assignment list 1 $
label1:
$ statement list $
continue_label:
$ assignment list 2 $
bra label1;
break_label:
```

Table 12: The for statement

PL/68K

(Continued from page 33)

is evaluated, and control either falls through to the *then* clause or a jump is made to the end of the statement.

Similar code is generated when the *if* statement contains an *else* clause. The Boolean expression is evaluated, and control either falls through to the *then* clause or a jump is made to the *else* clause. A BRA (branch always) instruction following the *then* clause skips around the *else* clause.

The code generated for the *while* statement, shown in Table 10, page 34, might be a little controversial. The first instruction is a branch to the end of the loop so that the loop test occurs at the bottom. This produces the fastest code unless the *while* loop is executed less than once on average. In the rare cases in which this jump is unwanted, the programmer must simulate the loop in some way.

Notice the labels called *continue_label* and *break_label*. These are used as target labels for the *break* and *continue* statements. In other words, within a *while*, *do*, or *for* statement, the effect of a *continue* instruction is to generate a branch to the *continue_label* defined for that statement. Similarly, a *break* statement generates a jump to the appropriate *break_label*. As in C, you can also use the *break* statement inside a *switch* statement.

The *while* statement generates different code if the Boolean expression is a nonzero constant. This is a common idiom in C, and the definition of PL/68K ensures that there is no time penalty for using it.

The code for the *do* statement, shown in Table 11, page 34, is similar to the code produced by the *while* statement. The code for the *for* statement (see Table 12, page 34) is more interesting. If the loop test in a *for* statement is non-trivial, the code for it appears at the bottom of the loop. Note also that the syntax of the *for* statement is more restricted than in C.

The *switch* statement, shown in Table 13, page 37, generates a jump table—i.e., a table of addresses. Code is generated that jumps through that table to the proper *case* statement, based on the contents of a register. Note that the *switch* statement destroys this register.

It is sometimes better to generate a sequence of tests rather than a table jump, but the *case* statement always generates a table jump. Remember, each language construct in PL/68K stands for a particular sequence of code—if you want a sequence of tests, use a sequence of *if* statements; if you want a table jump, use a *switch* statement.

Many compilers generate jumps to jumps when they translate structure statements, but the definition of PL/68K requires that all jumps to jumps (and jumps to return statements) be eliminated. The assembler can do this in several ways. For instance, if the assembler creates a parse tree for an entire function before any code is generated, it's easy for the code generator to look at the target of any jump to see if it is another jump or a return instruction. Alternatively, the assembler can use a standard peephole optimizer.

Function Calls

Functions in PL/68K can have formal parameters and *lc*

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BULLETIN BOARD - 7PM to 7AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products

SERVICE: FREE NEWSLETTER

Software development and AI on micros: trends, forecasts, controversies, innovations, and techniques. Plus announcement of 80 NEW tools. CALL for "Newsletter Packet."

RECENT DISCOVERY

dBASE to C translator: dBX - no royalties, addon ISAM,

MSDOS \$ 350
Source \$1000

AI - Expert System Dev't

ExpertEASE - Inductive. PC DOS Call
EXSYS - All RAM, Probability. Why.
Trees, Solid, files, popular PC DOS \$359
INSIGHT 1 - Probabilities, required
thresholds, menus, fast PC DOS \$ 95
INSIGHT 2 - adds backward, forward, par-
titions, dB2, lang. access. PC DOS \$449
Others: APES (\$359), Advisor (\$949),
ES Construction (\$100), ESP (\$845),
Expert Choice (\$449), more.

AI - LISP

List Our
GC LISP - "Common", rich. 495 Call
Interpreter - Interactive Tutorial 695 649
LARGE Model - 2 to 15 meg. 1190 1045
Compiler and LM Interpreter
TLC LISP - "LISP-Machine" - like, all
RAM, classes, turtle graphics, 8087,
compiler. CPM-86, MSDOS \$235
WALTZ LISP - "FRANZ LISP" - like, big
nums, debug, CPM-80, MSDOS \$149
Others: ExperLISP (\$439), IQ LISP
(\$155), TransLisp-PC (\$75),
BYSO (\$125), MuLISP-86 (\$199)

AI - PROLOG

ARITY PROLOG - full, debug, ASM, C,
virtual. Compiler \$1950 MSDOS \$495
MPROLOG - Rich syntax, editor, segment
work space, portable. PC DOS \$725
Prolog-86 - Learn Fast. Standard,
tutorials, samples MSDOS Call
Others: Prolog-1 (\$359), Prolog-2 (\$1895),
MicroProlog (\$229), Prof. MicroProlog
(\$359).

Editors for Programming

BRIEF Programmer's Editor - undo,
windows, reconfigure PC DOS Call
C Screen with source 80/86 \$ 75
EMACS by UniPress - powerful,
multifile, windows, DOS, MLISP.
programming. Source: \$949 \$299
Entry System for C - Bellesoft PC DOS \$325
FirsTime by Spruce - Improve
productivity. Syntax directed for Turbo
(\$69), Pascal (\$229), or C (\$239)
PMATE - power, multitask 80/86 \$159
VEDIT - well liked PC DOS \$119
XTC - multitasking PC DOS \$ 95

Feature

Paragon PASCAL - for performance:
extensions like "packages"; "Iterators";
5 memory models; 64 bit 8087; strings.
Space vs. speed optimization options.
MSDOS \$895

C Language - Compilers

BDS C - solid value, fast CPM80 \$125
C86 by CI - 8087, reliable MSDOS Call
Lattice C - from Lifeboat MSDOS \$289
Lattice C - from Lattice MSDOS \$339
/Consulair Mac C w/toolkit MAC \$299
Megamax - tight, full MAC \$239
Microsoft C 3.0 - new, tight. MSDOS \$259
Q/C 88 by CodeWorks - **Compiler source**,
decent code, cross/native MSDOS \$295
Williams - source debug. MSDOS \$399
Wizard C - Lattice C compatible,
full Sys. III, lint, fast. MSDOS \$399

C Language - Interpreters

C-terp by Gimpel - full K & R, .OBJ
and ASM, large progs. MSDOS \$249
INSTANT C - Source debug, Edit to
Run-3 seconds MSDOS \$399
INTRODUCING C - Interactive C to
learn fast, tutorial PC DOS \$115
RUN/C - improved MSDOS \$109

C Libraries - General

Blaise C Tools 1 (\$109), C Tools 2 \$89
C Food by Lattice - ask for source \$119
C*LIB by Vance \$125
C Utilities by Essential - 300+ \$149
Greenleaf Functions - portable & ASM \$149
Polytron - for Lattice, ASM source \$ 99
Software Horizons - Pack 1 \$129

C Libraries - Applications

COMMUNICATIONS: Asynch Mgr \$175
Greenleaf - full, respected \$149
Software Horizons - Pack 3 \$139
FILES: Btrieve - multilanguage \$199
C Index by Trio - full B + Tree,
vary length field, multi compiler
/File is object only \$ 89
/Pro is partial source \$179
/Plus is full source \$349
C Tree by Faircom - source, port. \$349
dbcISAM by Lattice for dB2 or 3 \$219
dbVISTA - full indexing, plus optional
record types, pointers. Network.
Object only - MS C, LAT, C86 \$179
Source - Single user \$459
Source - Multiuser \$929

Ask about Atari ST, Amiga

Note: All prices subject to change without notice.
Mention this ad. Some prices are specials.
Ask about COD and PO's. All formats available.

Call for a catalog, literature, and solid value

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339

Mass: 800-442-8070 or 617-826-7531 1085

C Support - Systems

C Debug - Source debuggers - by
Complete Soft (\$269), MSD (\$149).
C Sharp - well supported, Source,
realtime, tasks MSDOS \$600
C Sprite Debugger by Lattice \$149
C ToolSet - DIFF, xref, source \$135
PC Lint - full C program checking
and big, small model. All C's \$119

Low Cost Languages

ECO C/88 by Ecosoft \$ 50
Introducing C - Step by step training \$109
TransLisp-PC - "Common Lisp", tutorial,
graphics, 230 functions, samples \$ 75
Modula 2 by ITC - Windows, tight \$ 80
Prolog-86 - enhanced, DOS, Edit Call
Quick BASIC by Microsoft - Compile
BASICA, Link \$ 79
Snobol4+ by CatSpaw - Strings \$ 85
Turbo Edit/Assembler \$ 85

TURBO PASCAL and SUPPORT

BORLAND: Turbo 3.0 \$ 49
3.0 with 8087 or BCD \$ 79
3.0 with 8087 and BCD \$ 85
Turbo Graphix - graphs, windows \$ 39
Turbo Toolbox or Editor \$ 55
Turbo Tutor \$ 29
TURBO . . . Asynch by Blaise, full \$ 89
MetaWindow by Metagraphics \$ 49
Power Tools by Blaise - library \$ 89
Power Utilities - profiler, pp \$ 89
Professional - interrupts, macros, \$ 50
OTHERS: FirsTime (\$69), Screen
Sculptor (\$99), Pascal Pac (\$100),
Tidy (\$45), Multi Halo (\$95).

Fortran & Supporting

Forlib+ by Alpha - graph, comm. \$ 59
MACFortran by Microsoft - full '77 \$239
PolyFortran - xref, pp, screen \$149
Prospero - '66, reentrant \$390
RM Fortran - enhanced "IBM Ftn" \$429
Scientific Subroutines - Matrix \$149
Strings and Things - registers, shell \$ 59

MultiLanguage Support

Advanced Trace 86 Symbolic, rewrite
Assembler \$149
Btrieve/N (\$469), single user \$199
Codesifter - executive profiler \$109
LMK Make by Lattice \$159
MultiHalo - full \$199
Panel - Screens, windows \$239
Periscope II symbolic debugger \$129
PFinish - Profile by line, routine \$299
PLink-86 - 32 levels, overlays \$289
PolyLibrarian - Manage .OBJS \$ 89
TexSys - Source code control \$ 89

8080
Z80
6809

AMX

8086
8088

Now
68000

Real-Time Multitasking Executive

- No royalties
- Source code included
- Fault free operation
- Ideal for process control
- Timing control provided
- Low interrupt overhead
- Inter-task messages

Options:

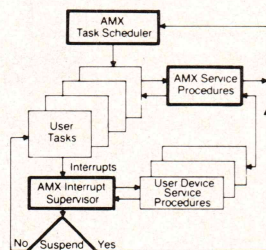
- Resource Manager
- Buffer Manager
- Integer Math Library

Language Interfaces :

C Pascal
PL/M Fortran

DOS File Access :

CP/M-80
IBM PC DOS



AMX is TM of KADAK Products Ltd.
CP/M-80 is TM of Digital Research Corp.
IBM, PC DOS are TM of IBM Corp.

AMX for 8080	\$ 800 US
8086	950
8089	950
68000	1600
Manual (specify processor)	75



KADAK Products Ltd.

(604) 734-2796

Telex: 04-55670

206-1847 W. Broadway, Vancouver, B.C., Canada V6J 1Y5

Circle no. 215 on reader service card.

UNIX/XENIX COMPATIBILITY!



C FUNCTION LIBRARY

C-LIB incorporates over 200 routines to extend the capabilities of C on the IBM PC. Versions are available for the Lattice, Microsoft, and DeSmet (C Ware) C compilers. All libraries are compatible with PC-DOS 2.0+ operating systems.

Featuring:

- UNIX-Compatible "CURSES" Screen/Window Processing Library
- Buffered, Interrupt-driven, Asynchronous Communications Library
- Powerful String Functions
- 8087/Microsoft Floating Point Conversion

C-LIB SOFTWARE AND MANUAL PACKAGE \$195.

Please call or write for additional information.
DEMO DISKETTE AVAILABLE, please include \$5.

VANCE info systems

2818 clay street • san francisco, ca 94115 • (415) 922-6539

IBM & PC-DOS are trademarks of International Business Machines Corp. Lattice is a trademark of Lattice Inc. Xenix & Microsoft C are trademarks of Microsoft Inc. UNIX is a trademark of Bell Labs. C-LIB is a trademark of VANCE info systems.

Circle no. 154 on reader service card.

PL/68K

(Continued from page 34)

cal arguments, just as in C. The code shown in Table 14, page 37, is generated by function calls. Code is generated to push all arguments on the stack, a JSR (jump to subroutine) instruction is generated, and, if necessary, an ADD instruction is generated to pop arguments off the stack. One long word is always reserved on the stack for the first argument, which shortens the calling sequence when there are less than two arguments.

The ADD instruction can be eliminated by having the called program, instead of the calling program, pop the arguments off the stack, but the sequence shown in Table 14 is the fastest. If you eliminated the ADD instruction and pushed the arguments in the same way (that is, above the return address), the called program would need to do much more work to pop off its arguments. You could also push the actual arguments below the return address, but that way actually increases the length of the calling sequence. In order to push arguments below the return address, you would have to use an instruction such as *move arg,n(sp)*, which is 2 bytes longer than *move arg,-(sp)*.

Unlike standard C, PL/68K does specify the order in which arguments are pushed, namely in reverse order. Thus, a function that takes a variable number of arguments—*printf()* for instance—will find its first argument on the top of the stack.

Of course, it is often best to pass arguments in registers, but PL/68K doesn't need a separate mechanism to do this. Suppose you have a function called *g()* whose two arguments are passed on the stack. To change *g()* so that it will take its arguments in registers, you just define the following macro

```
#define g(a,b)d0=a; d1=b; g1()
```

and change *g's* name to *g1*. Notice that a statement such as

```
if (..) g(x,y);
```

cannot cause problems in PL/68K because you must write

```
if (..) {g(x,y);}
```

instead. (If braces were omitted, after macro expansion, the code would be

```
if (..) d0=a;d1=b;g1();
```

and only the assignment *d0=a* would be part of the *if* statement.)

This macro might generate redundant code. Suppose it were called with *d0* as the first argument, for instance. The macro would expand to *d0=d0* and generate the instruction *move d0,d0*. To handle that problem, the PL/68K assembler eliminates redundant moves. If you must generate such a redundant move for some reason, use a pseudofunction. Pseudofunctions are never second-guessed by the assembler.

Syntax

```
switch ( reg ) {
case constant 1: statement list 1;
case constant 2: statement list 2;
...
case constant n: statement list n;
default:      default statement list;
}
```

Code generated

```
if (reg < min || reg > max) {
    goto default_label;
}
else {
    $ goto the routine whose address is at table [reg] $
}

table:
dc.l label i_min;
dc.l label i_min+1;
...
dc.l label i_max-min+1;

label 1:
$ statement list 1 $
label 2:
$ statement list 2 $
...
label n:
$ statement list n $
default_label:
$ default statement list $
break_label:
```

Table 13: The switch statement

No arguments

```
jsr    function
```

One argument

```
move   arg, (sp)
jsr    function
```

Two or more arguments

```
move   argn, (sp)
move   argn-1, -(sp)
...
move   arg2, -(sp)
move   arg1, -(sp)
jsr    function
add     # size of all arguments except argn, sp
```

Table 14: Code generated for function calls

Entry

```
movem.l all_local_registers, -(sp)
link     an, # - size of local auto variables - 4
```

Exit

```
unlk     an
movem.l  (sp)+, all_local_registers
rts
```

Table 15: Code generated for entry/exit of A_n-based functions

C Source Code

RED

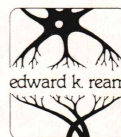
Full Screen Text Editor

IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
- RED is easy to use for writers or programmers. RED's commands are in plain English.
- RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
- RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
- RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.

RED: \$95

Manual: \$10



Call or write today for
for more information:
Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5 1/4 inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.

Circle **no. 90** on reader service card.

Release 2.0

C-INDEX™

Variable Length Record Management For C

C-INDEX is a state-of-the-art data management function library for C programmers. Ideal for all data and text based applications. No other package can give you the performance, capability, and portability of C-INDEX. To make sure you are a satisfied customer, we offer a 30 day money-back guarantee. Ask us about it.

- variable length data storage with B+Tree indexing
- high performance, easy to use
- large and small models, fully transportable source
- IBM PC format: Lattice, Microsoft, C86, others
- Macintosh format: Consulair, Manx, others

C-INDEX/FILE (\$99) Object code package.

C-INDEX/PRO (\$195) Partial source code, no royalties.

C-INDEX/PLUS (\$395) Complete transportable source code, no royalties.

Trio Systems

2210 Wilshire Blvd., Suite 289
Santa Monica, CA 90403
213/394-0796

Circle **no. 150** on reader service card.

Accessing Variables Within Functions

Now that I've covered the passing of arguments to a function, I'll explain how the function gets hold of those arguments. This is a complicated subject, so before getting involved in some messy details, let's handle the easy cases.

First, PL/68K keeps its hands off all registers declared outside any function. These global registers can be accessed from within functions, but PL/68K never generates code to save or restore them. Consequently, you can prevent PL/68K from interfering with any register simply by declaring that register outside a function. By the way, the register keyword is not valid outside functions, but that does not prevent you from declaring register variables anywhere you wish, either in C or in PL/68K. For instance, you can declare *a0* to be global simply by saying *char *a0;* outside any function.

Second, except for these global registers, all registers used in a function are saved on entry and restored on exit from the function. The assembler uses the MOVEM.L (move multiple register, long) instruction for this purpose. Accessing register variables declared in a function is, of course, easy.

Third, if a local variable is declared to be static, memory is allocated to that variable in static memory, not on the stack. No extra code is needed to access that variable, either on function entry or exit.

Static internal variables are not very useful; because the values of static internals are retained between invocations of a function, static internals are destroyed by recursive function calls. Also, on many machines (including the 68000) accessing a variable in static memory is more expensive than accessing a "local auto" variable—that is, a local stack variable. At any rate, generating code for static internal variables is straightforward and is not affected by the following complications.

Functions need to access two kinds of nonregister variables: formal parameters and local auto variables. Both types are allocated on the stack. You have three choices

for how PL/68K generates code for these stack variables. You make your choice using one of three pseudofunctions: *base(A_n)*, *base(sp)*, or *nobase()*. If none of these appears in a function, the default is *base(sp)*.

First, you can have PL/68K access stack variables via an address register that is different from the stack pointer, as shown in Table 15, page 37. If the *base(A_n)* pseudofunction appears anywhere in the function (where *A_n* is any address register except the stack pointer, *a7*), PL/68K generates a LINK (link and allocate) instruction on function entry and an UNLK instruction on function exit. All stack variables are accessed via the address register named in the *base(A_n)* pseudofunction.

Second, you can have PL/68K access stack variables via the stack pointer, as shown in Table 16, page 38. The assembler generates this kind of code if the *base(a7)* or *base(sp)* pseudofunction appears anywhere in the function. The code generated for *sp*-based functions is more complicated, but more efficient, than that for *A_n*-based functions. If an *sp*-based function contains no function calls, the stack pointer is not incremented on function entry and local auto variables are accessed using positive offsets from the stack pointer. If the function does contain other function calls, however, space is reserved for local auto variables by incrementing the stack pointer on function entry, and local auto variables are accessed using negative offsets from the stack pointer.

Third, you can access stack variables by hand. If the *no_base()* pseudofunction occurs anywhere in the function, no code is generated on function entry or exit, declarations of stack variables are ignored, and no explicit references to stack variables are permitted. The *no_base()* pseudofunction is useful when you must play some kind of game with the stack.

Sp-based functions are somewhat dangerous. If the stack pointer is changed using a pseudofunction, the offsets used to access stack variables are going to get out of sync. To handle this problem, several pseudofunctions, shown in Table 17, page 38, allow you to indicate that the offsets should be changed.

The *push(arg)* and *pop(arg)* pseudofunctions are equivalent to the *move(arg, *-sp)* and *move(*sp+, arg)* pseudofunctions, but in addition, they tell the PL/68K assembler to adjust the offsets used to access stack variables in *sp*-based functions. The *addsp(n)* and *subsp(n)* pseudofunctions are equivalent to the *addi(n, sp)* and *subi(n, sp)* pseudofunctions, but again they tell the assembler to adjust offsets. Finally, the *adjsp(n)* pseudofunction generates no code but tells the assembler to adjust the offsets.

If one or more functions called within this function

Entry

```
movem.l  all_local_registers, -(sp)
suba     # size of local auto variables + 4, sp
```

Exit

```
adda     # size of local auto variables + 4, sp
movem.l  (sp)+, all_local_registers
rts
```

If no function called within this function

Entry

```
movem.l  all_local_registers, -(sp)
```

Exit

```
movem.l  (sp)+, all_local_registers
rts
```

Table 16: Code generated for entry/exit of *sp*-based functions

Pseudofunction	Meaning
<i>base(a_n)</i>	use <i>a_n</i> basing for stack variables
<i>base(sp)</i>	use <i>sp</i> basing for stack variables
<i>nobase()</i>	access all stack variables "by hand"
<i>push(arg)</i>	move <i>arg</i> , <i>-(sp)</i> and adjust stack offsets
<i>pop(arg)</i>	move <i>(sp)+</i> , <i>arg</i> and adjust stack offsets
<i>addsp(n)</i>	adda # <i>n</i> , <i>sp</i> and adjust stack offsets
<i>subsp(n)</i>	suba # <i>n</i> , <i>sp</i> and adjust stack offsets
<i>adjsp(n)</i>	adjust stack offsets

Table 17: Pseudofunctions for stack operations

Free CompuView Software!
Call (313) 996-1299 for details.

VEDIT PLUS[®]

Multiple File Editor

Word Processor

RECOMMENDED

Come and get it. The editor you've heard so much about. The editor that has been recommended to you for the last five years by BYTE, InfoWorld, Microcomputing, PC, Programmers Journal, Sextant, EDN, Jerry Pournelle, Peter Norton and numerous other reviewers.

'VEDIT is a lightning fast text editor with all the commands of a slick word processor...a text oriented programming language enables you to perform tasks impossible with a standard word processor. A fantastic product.'

The Whole Earth Software Catalog, 1985.

'VEDIT is fast, functionally rich and configurable to your whims. Its programming ability lets its usage stretch as far as your imagination will allow.'

The OMNI Complete Catalog of Computer Software, 1985.

Now there's new VEDIT PLUS. It does it all. Program development. Document preparation. Word Processing. Convert WordStar files, edit dBASE source files, process mainframe files. Whatever your application, VEDIT PLUS can handle it.

VEDIT PLUS is the choice of thousands of engineers, writers and programmers who demand the most powerful text editing software.

Expect more from VEDIT PLUS. It's available for all MS-DOS, PCDOS, CP/M-86 and CP/M-80 computers. List price \$225. Educational / corporate site licensing and current user discounts available.

VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size. Do 'cut and paste' operations, edit text or develop macros.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K
- MS-DOS, PCDOS pathname and CP/M user number support.
- Horizontal scrolling (edit spreadsheets easily).
- Customization - determine your own keyboard layout, support any screen size, any computer, any CRT terminal

EASY TO USE

- Interactive on-line help. Create your own on-line help with menus for compilers, style guides, non-computer subjects.
- On-Line integer algebraic calculator.
- 'Undo' command.
- Single key search and selective replace. Search with wild cards and pattern matching.
- Keystroke macros - create your own on-line editing functions.
- Print any portion of text or file.
- Excellent indexed documentation. Includes new tutorial.
- Highly optimized for IBM PC, PC XT or PC AT.

PROGRAM DEVELOPMENT

- Automatic Indent/Undent for 'C', PL/I or PASCAL.
- Conditional lower to upper case conversion. Change at ';' or other character.
- Optional 8080 to 8086 source code translator macro.

WORD PROCESSING

- Word wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Recognizes graphic, foreign and special character sets.

MACRO PROGRAMMING LANGUAGE

- 'IF-THEN-ELSE', looping, conditional branching, user prompts, keyboard input, algebraic expressions and variables.
- Simplifies complex text processing, formatting, conversions and translations.
- Edit multiple files automatically - perform numerous search/replace in several files without user intervention.
- Complete TECO capability.
- Free Macros: • Compare two files and merge work done by two people • Sort mailing lists • Print formatter • Replace command prompt with an easy to use menu.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc.
WordStar is a registered trademark of MicroPro, International.
dBASE is a registered trademark of Ashton Tate.
MS-DOS is a registered trademark of Microsoft.
CP/M is a registered trademark of Digital Research.

CompuView[®]

These pseudofunctions have no effect on A_n -based functions.

Summary

To summarize the strengths and weaknesses of PL/68K, the syntactical restrictions that make PL/68K a strict subset of C are listed as follows:

- All operand/operator combinations must correspond to an instruction in the 68000 instruction set.
- Functions neither have types nor return values.
- The array operator [] is eliminated, but arrays may be declared.
- Assignment statements are severely restricted.
- Statement lists must be surrounded by curly brackets.
- There are no floating points, or double constants, or operators.

There are no additions or changes to C syntax. PL/68K and assembly language are semantically identical; the advantages of PL/68K over assembly language are all syntactical:

- C syntax makes programs easier to read.
- C syntax eliminates many common coding errors.
- Far fewer visible labels are required.

PL/68K is not the successor to C nor is it superior to C for most programming projects. The advantages of PL/68K over C apply only in limited circumstances, but when performance is paramount, PL/68K stands out:

- All machine resources and instructions are available.
- Global allocation of registers is possible.
- Total control over generated code is possible.
- Generated code is smaller and faster.

Figure 2, page 42, shows the relationship between PL/68K and its two parent languages. PL/68K is only a subset

of C; not all of C is included. On the other hand, C must be augmented by the ops library in order to simulate all the machine resources of assembly language.

Listing One, page 101, shows an example of a PL/68K function. This function finds a name in a symbol table and returns information about that name in registers. Listing Two, page 101, shows the code generated by the PL/68K assembler for that function.

Conclusion

PL/68K started life as a C-like assembly language for the 68000 chip. High-level assemblers are not new—a paper by Niklaus Wirth (*Journal ACM* 15, January 1, 1968) described a high-level assembly language for IBM 360 machines.

My early thinking about PL/68K was influenced by Wirth's paper and by the register allocation and code selection principles. At that stage, I was interested in using compiler technology to create better assemblers. I saw no particular reason to make PL/68K a subset of C.

I felt dissatisfied with the initial version of PL/68K; it was doomed to be "just another computer language." Not wanting to add another minor dialect to the Babel of computer languages, I decided to make PL/68K's syntax identical to C's. This was a lucky decision.

From the first, PL/68K had to be able to name all machine resources, including machine instructions and assembly-language pseudo-ops. Early versions of the language allowed "raw" lines of assembly code to be interspersed with C-like lines of code. Although this approach had some merit, my decision to make PL/68K's syntax compatible with C convinced me to use functional notation to represent assembly-language features. This change was also fortunate.

I began to speculate about what would happen if a PL/68K program were compiled with a C compiler. I asked myself, suppose the program produced by the C compiler and the program produced by the PL/68K assembler were semantically equivalent? Suddenly, PL/68K became not just another assembly language but a new way of using C.

The principle of semantic equivalence guided a rede-

The same code may be run through two different translators.

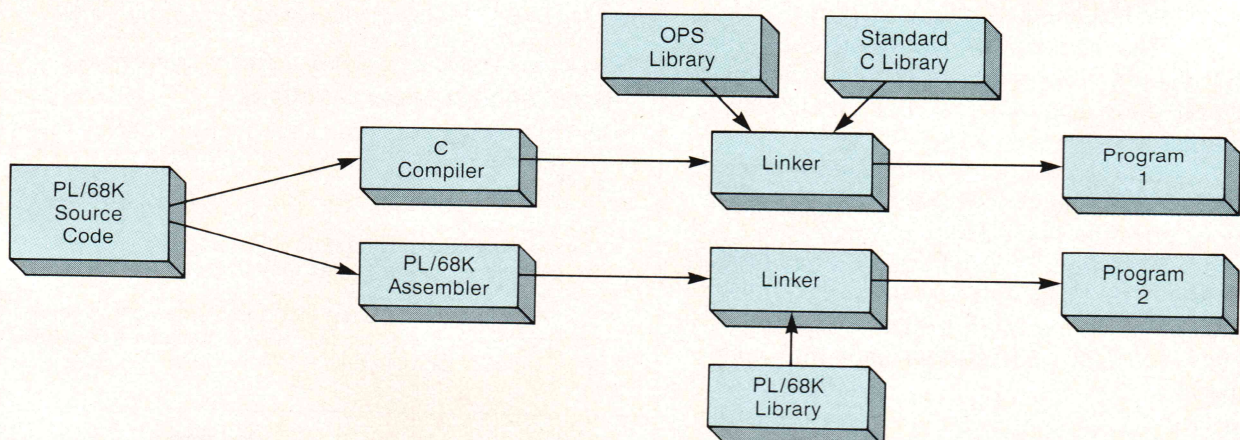
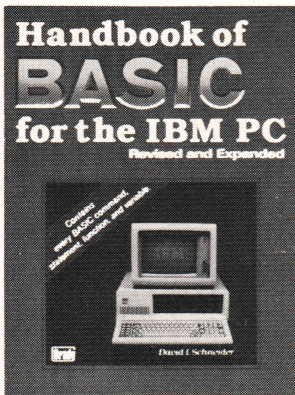


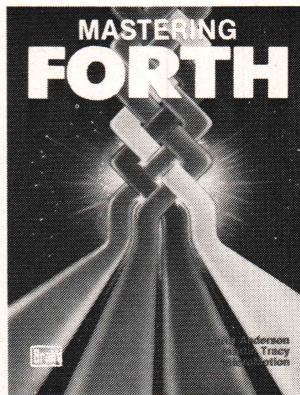
Figure 1

BRADY Speaks Your Language

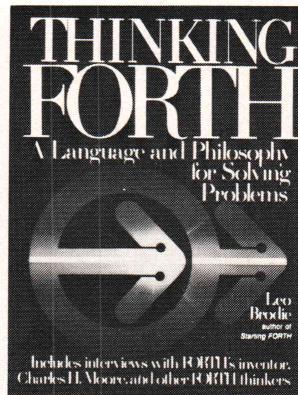
We can help you buff up your BASIC... Put a finish on your FORTH...
Conquer C or COBOL... Learn how to LISP... Succeed with Assembler...
And more! Just call toll-free or use the coupon to order today!



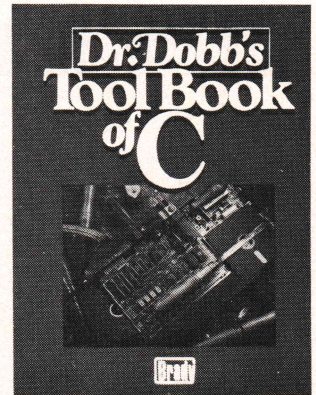
1. PC magazine calls it: "A truly remarkable book... A treasure trove of useful programming information for the IBM PC." \$22.95



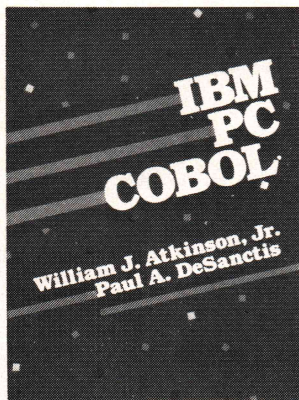
2. A step-by-step tutorial for the high-level, stack-oriented FORTH-83 standard. \$17.95



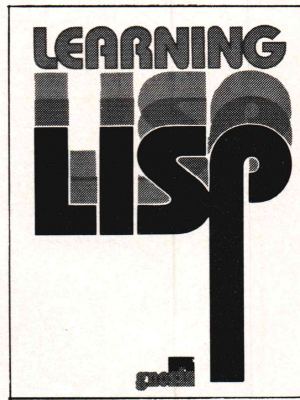
3. The FORTH authority, Leo Brodie, illustrates the elegant logic behind FORTH and shows how to apply specific problem-solving tools to software. \$15.95



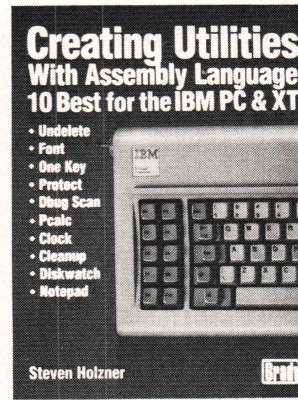
4. The best C articles from the highly respected *Dr. Dobb's Journal* dealing exclusively with C language and programming techniques. \$24.95



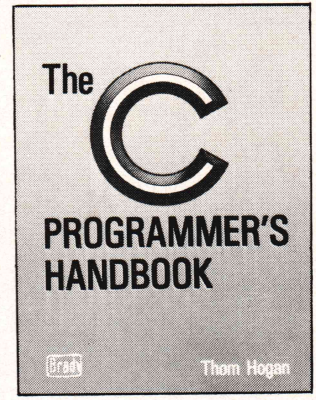
5. The only book of its kind to focus on the differences between micro and mainframe COBOL—and that includes invaluable, fully coded COBOL programs for the IBM PC. \$19.95



6. A new guide to LISP (List Processor), the much-talked-about language of artificial intelligence. \$14.95



7. For the more advanced user familiar with Assembly language, here's an opportunity to unleash the power of 10 DOS-enhancing programs. \$19.95



8. The definitive desktop reference for anyone using the C language—a virtual clearinghouse for C language information. \$19.95

Circle no. 219 on reader service card.

Now at your book or computer store.
Or order toll-free today:

800-624-0023

In New Jersey:
800-624-0024



COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall,
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below.
(Payment must be enclosed; or, use your charge card.) Add \$1.50 for postage and handling.
Enclosed is check for \$_____ or charge to

☐ MasterCard ☐ VISA.

Acc't # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

(New Jersey residents, please add applicable sales tax.)

G-LDDJ-AE(1)

1 (0-89303-510-6)
5 (0-8359-3051-3)

2 (0-89303-660-9)
6 (0-13-527813-9)

3 (0-13-917568-7)
7 (0-89303-584-X)

4 (0-89303-599-8)
8 (0-89303-365-0)

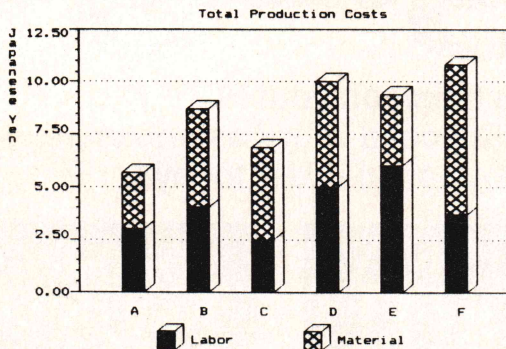
GRAF 3.0

the complete
BUSINESS and SCIENTIFIC
printer graphics program

- display floating point data directly from spreadsheets, data bases, word processors, and programming languages (or the keyboard) in a wide variety of bar, pie, line, and scatter plots
- menu driven operation supports automatic graph scaling, labeling, and legend creation

\$49⁹⁵ **\$69⁹⁵**
CP/M-80 MS-DOS / PC-DOS

- plot and group up to 6 different variables on a single graph, distinguished by up to 14 different fill-in patterns and 8 different point-plotting symbols
- add up to 5 different-density grid lines, and choose from a wide variety of numerical labeling options



GRAF 2.0 Update Policy: Returning your original GRAF 2.0 disk to MSC entitles you to \$20.00 off the above prices.

TERMS: We ship via first class mail. The above prices include domestic shipping and handling. Orders outside USA require additional \$5.00 for postage. N.Y. residents add state sales tax. When ordering you MUST state your computer and printer make and model. We support MS-DOS (PC-DOS) version 2.0 or later on computers with at least 192K RAM, and CP/M-80 version 2.2 or later on Z80 computers (other than modified Apples) supporting a TPA of at least 54k (requires 96k of RAM). Most soft-sector disk formats are available. (If you can read several formats, please send us a list.) GRAF 3.0 works with any printer fully compatible with one of the following: Epson FX, RX, LX, MX (with GRAFTRAX), or LQ-1500; C. Itoh Prowriter; NEC 8023A; Star Micronics Gemini 10K, 15K, SG-10, SG-15; IBM Graphics Printer; Okidata 192, and earlier Okidata models equipped with the "IBM Plug 'n' Play" chips. (If you have an Okidata printer, other than the 192, the Plug 'n' Play chips are required.)

MSC

Microcomputer
Systems
Consultants

27 Forest Avenue Port Jefferson Station New York 11776-1820

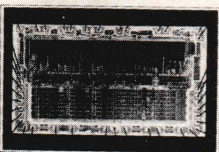
Circle no. 136 on reader service card.

FROM THE DEVELOPERS OF THE

65816 Microprocessor

The programming handbook you've been waiting for.

Programming the
65816 Microprocessor
Including 6502 and 65C02



David Eyes

- Outlines the programming strengths of the 65816, 6502, and 65C02.
- Includes a review of basic concepts, architecture, and logical operations.

Now available at your local bookstore, or call toll free 1(800)624-0023 to order your copy today. In New Jersey, call 1(800)624-0024.

Brady

0-89303-789-3/288 p/\$22.95

PL/68K

(Continued from page 40)

sign of the language; any construct that violated this principle had to go. In addition, I invented the ops library so that C programs could simulate 68000 machine instructions. Along with the ops library, the concept of pseudo-functions was born, and the language was complete.

PL/68K allows me to sidestep a question that has been haunting me ever since I started programming—whether to program in assembly language and accept the resulting inconveniences or to program in a high-level language and accept a final product that is larger and slower than it could be. PL/68K solves that dilemma.

You can design and write a program in C, keeping in mind the possibility that you will convert it to PL/68K eventually. You can write difficult parts of the program, or parts of the program that will not appear in the final version, using all of C's features. After you have debugged the C program, you can produce a PL/68K version of the program, if desired, by rewriting or excluding those parts of the program that use full C. You then test the PL/68K version and improve its performance as much as you require.

PL/68K hits precisely the right level of abstraction for systems programming. All the features of C allow you to design and code programs easily, but when you need to do low-level work, nothing stands in your way.

A final thought—you could transfer most of the syntax and all the design rules of PL/68K to a similar language for other machines. In this limited sense, PL/68K is a machine-independent assembly language—PL/68K programs are much more portable than programs written in traditional assembly language.

The relationship between PL/68K, C and assembly language.

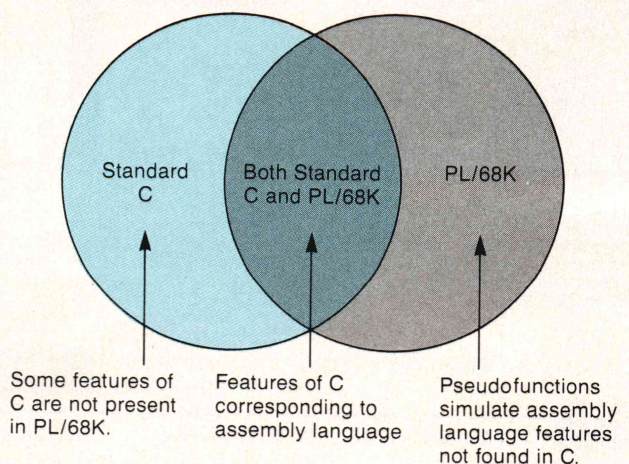


Figure 2

Note

A compiler for the PL/68K language is available from the author.

Appendix: Why Expressions Are Restricted in PL/68K

The outline of the argument is as follows. The register allocation rule conflicts with C's precedence rules (which govern the order in which operators are applied) and C's rules for converting operands from one type to another (known as the "usual arithmetic conversions" in the *C Reference Manual*). Thus, by the principle of semantic equivalence, all constructions in PL/68K that involve C's precedence and type conversion rules must be eliminated.

Consider a typical C expression, such as $a = (b * c) + (d * e)$. The register allocation rule says that this expression must be evaluated in location a , but that is not possible. At least two separate locations are required—one to hold the subexpression $(b * c)$ and another to hold $(d * e)$. In general, any expression involving parentheses may require more than one location to evaluate.

The only way to make a single location suffice for the evaluation of all expressions is to require left-to-right (or right-to-left) evaluation of operators. Alas, this changes the precedence of operators, except in cases such as $a = d * c + c$; which is evaluated left to right in C anyway.

Even such limited expressions cannot be allowed, though. C's rules for type conversions state that conversions are made during the evaluation of the right-hand side of the expression. Only after the right side is completely evaluated does the assignment take place. PL/68K has only one place to make those conversions, however—the left-hand side of the expression. The operands themselves cannot be converted because that would change their value. Thus, only one operand can be converted and only one operand can be allowed on the right side of an assignment statement.

DDJ

(Listings begin on page 101)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 2**.

Wizard C

Discover the powers of Wizard C
for yourself!

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

"Wizard's got the highest marks for support."

"The Wizard compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

Full Lint checking, six memory models, 8087 support, in-line assembly language, ROMable data support, full library source code. Cross-compilers are available on VAX/VMS and UNIX machines.

(617) 641-2379

Only \$450.

WIZARD
SYSTEMS SOFTWARE, INC.

11 Willow Court
Arlington, MA 02174



Circle no. 116 on reader service card.

RUN CP/M 80 ON IBM FAST!

Now your IBM PC or compatible can access the vast library of CP/M/80-2.2 software titles, and run them at lightning speed! Our revolutionary MICRUN/CPM non-emulating interface executes CP/M software on PC's by utilizing the power of NEC's newest microprocessor, the V-20. The V-20/30 chip replaces your PC resident 8088 or 8086 microprocessor chip, thereby providing 8 bit processing ability and actually improving it's 16 bit performance by up to 50%. If your PC is not already equipped with the NEC V-20 there is no problem. Simply purchase our MICRUN/CPM software package, and we will provide you with the V-20 chip for only \$45 dollars.

Features

- Incredibly fast-up to 8 MHz
- Ability to run CPM programs in color
- Cross sub-directory operations
- No expensive co-processor board req.
- Not an emulator
- Includes disk transfer utility for reading, writing, and formatting up to 70 different CP/M and MS-DOS disks.
- 16 logical/physical drives
- Includes RS232 communications program for transferring software to or from CPM system to PC.

Supports

Osborne* Kaypro* Zenith H-19* TeleVideo* Radio Shack* ADM3A* VT-52* Visual 210 adds Viewpoint* Hazeltine Espirit & 1510* Plus many more

ONLY \$99.95

Included in the above price: Interface software, disk transfer utility program, RS232 communications program, complete documentation. Not included is the NEC V-20 microprocessor chip which is available at a price of only \$45 dollars.



1-800-637-7226



Orders only.



Dealer Inquiries Invited

Trademarks: CP/M (Digital Research, Inc.), IBM (IBM Corp.).

Micro Interfaces Corporation
6824 N.W. 169th Street
Hialeah, Florida 33015
(305) 823-8088

Circle no. 110 on reader service card.

A Simple Multitasking Operating System for Real-Time Applications

by Nicholas Turner

The instruction set for the 68000 family is nearly orthogonal. This is important for a system on which a lot of assembly-language development work is to be done.

For the past year, we at Terra Nova Communications have been involved in a development project that requires a simple, fast, clean 32-bit microprocessor operating system. After a great deal of research, we were unable to find a commercial system that met our stringent requirements of extremely fast response time (even under a load of 20 users), low price (less than \$10,000 for both system and software), compact code size (we wanted a system kernel, including all the utility routines discussed in this article, that required less than 20K of object code), and simple programming of applications. After some brainstorming, we created a 68000 multitasking kernel that met and even exceeded our expectations of speed and compactness. Released from hardware requirements by our decision to write the kernel ourselves, we decided to use the VMEbus hardware configuration because of its standardization, complete hardware specification, relatively low price, ease of expansion, and the availability of lots of high-speed hardware devices. We were also impressed by the reliability and ease of use of the Eurocard connectors used with the VMEbus.

Why Not Use an Existing OS?

Our requirements for speed and compactness stemmed primarily from the need to handle a large number of I/O tasks over serial lines with-

out incurring large overheads for interrupt handling, disk access, and context switching—that is, we needed to be able to do significant amounts of I/O without slowing the system. Our kernel should eventually be able to handle 20 real-time users over serial lines at 1,200 bps, including full-speed block transfers, with no perceptible response-time delay and only minor slowing of the byte-transfer rates. We needed a system that would degrade gracefully; rather than pausing in midstream as one task takes over the system for an appreciable fraction of a second or as tasks are paged in and out, it should slow down gradually as the load increases, always providing steady and uniform output, even if it's at a reduced byte transfer rate. None of the commercial systems we examined had this property, nor were they able to handle the load required without significant degradation.

Examination of the code used in several of the commercial kernels we sampled showed some interest-

ing reasons for this: Most kernels contained code designed to handle all sorts of unlikely circumstances that might arise in an environment where you don't know what sorts of programs might be running. Not only did this code add significantly to the size of the kernel but it also slowed down the process of context switching between tasks in many cases. We needed the fastest possible context switch in order to guarantee that minimum system time was spent on this. Fortunately, we knew exactly which applications would be running on our system, and we were able to design a complete application/system interface to make application coding easy. Because we knew that all application code running under our kernel would be "polite" (would follow all the rules of the interaction between application and kernel) and that all source code would be available for debugging, we were able to dispense with a lot of the error-handling code usually present in commercial kernels.

We discovered that another contributing factor to the time required for a context switch was the magnitude of the context that was switched. In most of the systems we examined, all the machine registers were saved and restored, and full status information was saved, both of which took up a significant amount of processing time. As we'll explain later, we fixed this in a rather unorthodox way.

Further, many commercial kernels required the use of a memory manager chip and spent significant amounts of time paging users in and

Nicholas Turner, 10 McGranahan Court, Boulder Creek, CA 95006, (408) 338-9510

out to compensate for a small system memory. We opted against memory management, mainly because it solved no problems for us. We didn't need any sort of memory protection; in fact, one of the most important criteria was that all tasks must be able to quickly read and write data belonging to any other task or to the system itself. Also, our memory requirements were not large (minimum 512K, expandable to several megabytes) and because of the amount of code sharing between tasks, the structure of our data heap, and the heap's interaction with the disk system, there was no need for hardware paging of memory.

Why Program the Kernel in Assembly?

It was clear from the start that, in order to get the kind of performance we wanted from the system, the inner kernel had to be written in native code. Even compiled C or Forth would have had to be manually "tuned" in assembly source code form. Further, we could see several difficulties with compiled language—we needed to do so many "tricky" things to extract the last few cycles from the system kernel that writing it in compiled code was out of the question. By putting the entire kernel in machine code, we simplified the effort required to make major changes (it's all in the same language) and made possible a far more complete and integrated tuning.

Eventually we expect to put up at least a C compiler and a Forth interpreter for faster development, but for the moment all development is in assembly for speed and compactness. Because assembly was the language of choice, selection of the target processor was the next important issue.

Why Use the 68000?

The eventual goal of our project is to provide a responsive telephone dial-up system that is able to support up to 30 or 40 simultaneous calls without significant performance degradation. Such a task requires a truly powerful processor, even if the whole system is written entirely in native code. For several reasons, we have chosen the 68000 family of processors for our base hardware. The most important reason is that the instruction set is ex-

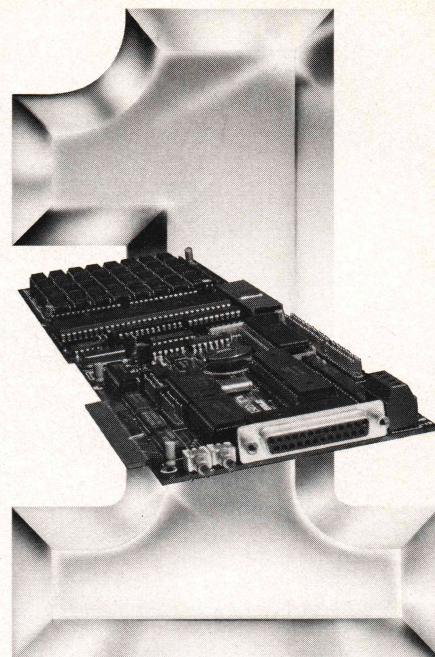
tremely versatile and powerful. It is in many ways a true 32-bit instruction set, although, unless you are using a 68020, you must put up with slightly slower memory access for 32-bit reads and writes because of the 16-bit data path.

The instruction set for the 68000 family is nearly orthogonal—that is, almost every instruction can be used with any of the 12 addressing modes. This is important for a system on which a lot of assembly-language development work is to be done because the programs become much easier to generate, read, and debug. Unfortunately, even the 68000 is not perfect. Several times we've encountered annoying restrictions—for example, we've often cursed our inability to do a PC-relative store.

The 68000 also has another advantage for assembly-language programmers: The memory architecture in its native mode, without the extras added by a memory management chip, is perfectly flat. That is, the address space is completely continuous from \$00 0000 all the way to \$FF FFFF—or \$FFFF FFFF if you have a 68020. For an assembly hacker, this is far more desirable than the segmented architecture required with the Z8000 or 80286 or with any 8- or 16-bit processor. For people writing in a high-level language, this is not an issue because they never deal directly with the memory at all. But for us, it's nice to be able to chop up that big address space in any way we like. As I'll explain later in this article, we have chosen to make it into an enormous heap and virtual disk area, thus making the fullest possible use of each and every byte.

Finally, after it became clear that the VME hardware bus was the best bet for our needs, the 68000 processor family was the logical choice because of the large number of 68000-oriented products available for the VME architecture.

In this article I'll describe briefly how our operating system fits together, and then I'll get to the fun stuff: the tricks and shortcuts we used to get such incredible performance out of the 68010 in our system. If you are already familiar with how a multitasking operating system fits together, you might want to skim down to the tricks and shortcuts.



**Number One
in Performance**
**68010/68000
Coprocessor for
IBM/AT/XT/PC-**
8/10/12.5mz No Wait States
\$1295⁰⁰ Qty. 1

FEATURES

- 1-2 MB RAM (1MB Standard)
- 16K-64K EPROM
- 2-8 Serial Ports
- Async/Sync/Bisync Communications
- Battery-backed Real Time Clock
- Battery-backed 2K-8K RAM
- 2 Parallel Ports
- 68881 Math Coprocessor
- Memory-mapped Dual-port BUS
- 3-9 Users Per Board (3 Standard)
- Up To 16 Boards Per AT/XT/PC
- Can Operate As Standalone Processor

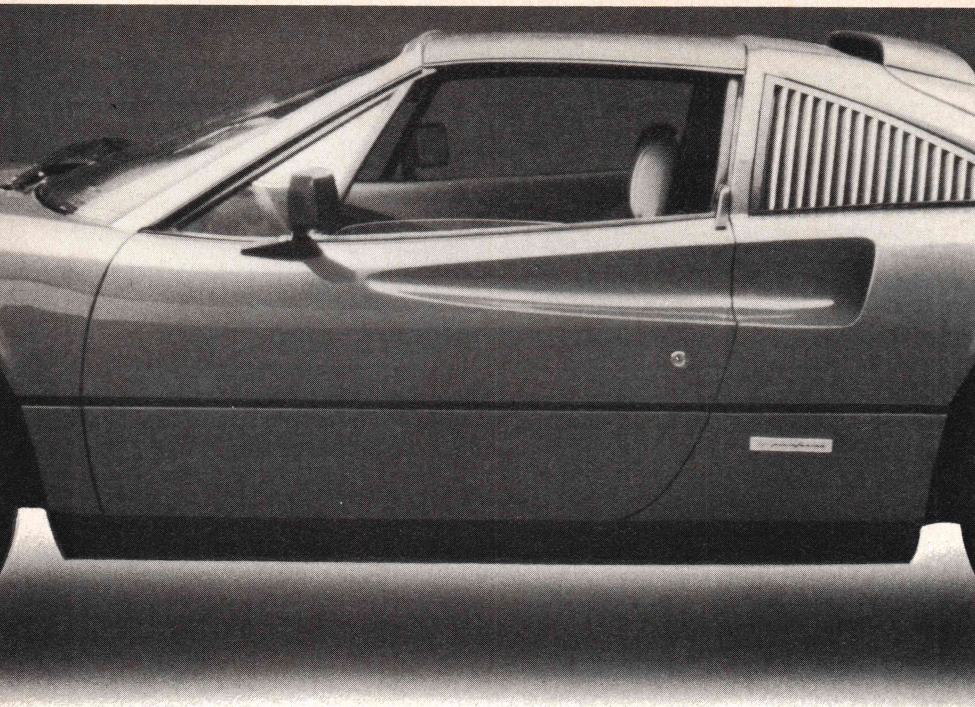
SOFTWARE

- OS9 (Powerful UNIX-like Multi-user OS)
- CPM/68K
- Software selectable OS including concurrent PC DOS/OS-9 or CPM/68K operation
- Support Module for IBM Graphics
- High-speed Local/Global Disk Caching
- Basic, Pascal, Fortran, C, and COBOL

IBM is a registered trademark of International Business Machines Corp. OS/9 is a registered trademark of Microvare Systems Corp. CPM/68K is a registered trademark of Digital Research Corp. MICROSOFT, MS-DOS, and Windows are registered trademarks of Microsoft. UNIX is a registered trademark of AT&T.


West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle **no. 174** on reader service card.



One Text Processor Beats Pmate™ The New Pmate.

No other PC/MS-DOS® text processor lets you edit files at the same time that you're compiling a program. Or, call up your editor while inside another application. Or, put long text processing jobs in the background and run other programs while they execute.

No other text processor offers you built-in macros that give you C and FORTRAN language-specific editing features. Brace matching. Movement, deletion, and copy by function. Keyword insertion. Error processing, and many more. Plus, since you get macro source code for all language-specific features, you can modify or add features as you wish.

And, no other text processor lets you use 1-2-3®-like menus, mouse pop-up menus, command-driven operation, or any combination of the above. With Pmate's 170-command macro language, you can program a single key to handle multiple command sequences. Call up other macros or common statements. Pass arguments to macros. Or, set up your

own personalized text processing system with all your favorite features.

Plus, you get full DOS 2.x path-name support. 100 numeric variables.

A number stack for storing intermediate results. Arithmetic and logical operations. Structured control statements. As well as numerous string, text and document manipulation functions.

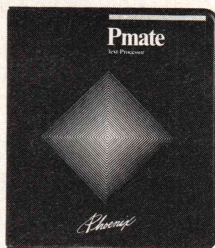
And, of course, you get standard Pmate features such as full-screen single-key editing. Auto-

matic disk buffering. Ten auxiliary buffers. Horizontal and vertical scrolling. A "garbage stack" buffer for retrieval of deleted text.

Pmate is priced at \$225 and is available for 8086-based micros running MS-DOS. A custom version is available for the IBM® PC, TI Professional™, Wang Professional™, and DEC Rainbow™.

Find out more about the new Pmate today! Call (800) 344-7200. In Massachusetts (617) 762-5030. Or, write.

Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062



PROGRAMMERS' PFANTASIES™
BY

Phoenix

MULTITASKING OS
(Continued from page 45)

Memory Structure

RAM memory is divided into two areas: the system zone and the heap, as shown in Figure 1 (page 47). The system zone begins at memory address 0 with the 68000 vector list and continues up from there. It's quite small and contains only a few data structures. Figure 2 (page 47) diagrams the structure of the system zone.

The 68000 vector list contains all the hardware vectors required for processing of interrupts and exceptions. It requires \$300 bytes. The data in the vector list is first set up by the initialization section and modified thereafter by the I/O manager as device interrupts are added or deleted.

Above the vector list is a small zone containing miscellaneous system variables and pointers. The time of day and date are kept here, along with pointers for the various linked lists maintained by the kernel, plus some other information that needs to be quickly accessible via absolute short addressing (the fastest way to get at a memory location from the 68000). Several I/O devices also have data here, where it can be accessed by interrupt routines without the overhead of following pointers through memory.

Following the system variable zone, we have a rather strange beast in today's world: an old-fashioned jump table! This table contains more than 100 absolute-long address mode JMP instructions at the moment—hundreds more are planned.

The next structure in the system zone is the task control buffer (TCB) table, which is an array of data structures that are linked via pointers into several doubly linked lists. Each task is associated with one TCB in the TCB array. When a task releases control to the next task, the context switcher reads the pointer from the outgoing task's TCB to get the address of the next task's TCB. This prevents unnecessary overhead while reading TCBs belonging to inactive tasks in the array because they are not part of the active task linked list. It also makes possible an extensible TCB array: If the primary array is full when another task is about to be spawned, the task manager can allocate a nonrelo-

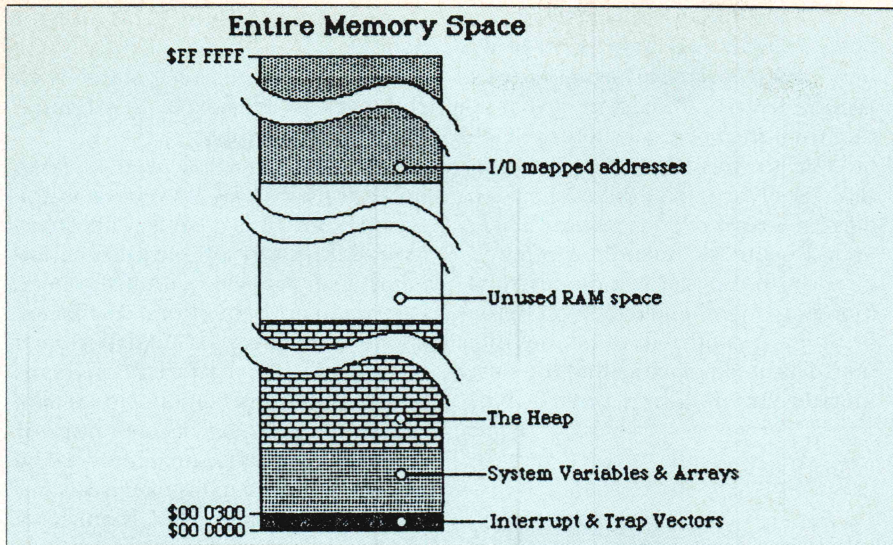


Figure 1: Basic organization of the 68000 memory space

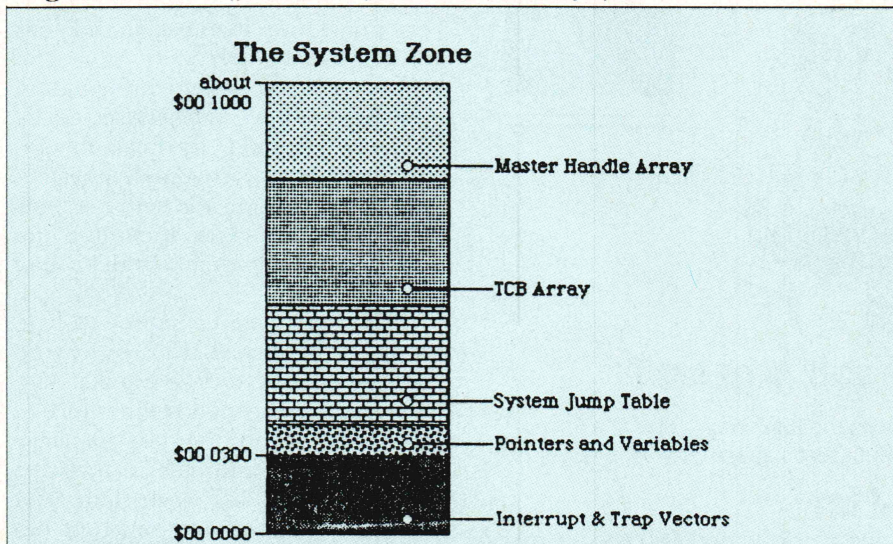


Figure 2: The system variables and arrays

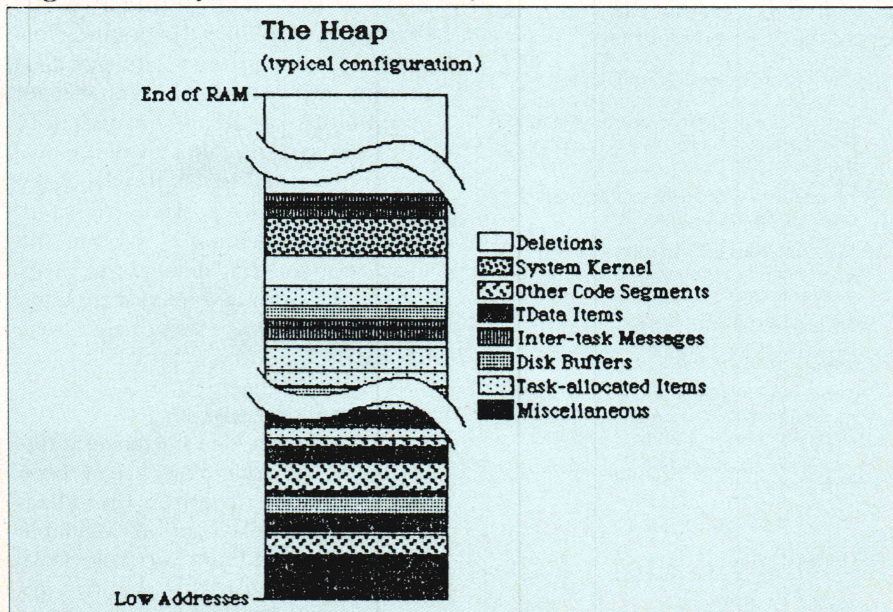


Figure 3: Typical structure of the heap



Number One In Performance

Hard Disk Intelligent VCR Backup for AT/XT/PC

FEATURES

- High speed microprocessor controlled backup (68000)
- Two channel interface
- Built in LAN channel
- Software control of most VCR functions including Fast Forward, Rewind, and auto backup using VCR timer capabilities
- Economical VHS or Beta formats



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
 East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
 Distributor: Telemarketing Services, Inc.
 1897 Garden Ave., Eugene, OR 97403, 503/345-7395
 Circle **no. 175** on reader service card.

catable item in the heap and continue the TCB array there.

Another extensible array follows the TCB table: The master handle array contains handles to relocatable heap items. (A handle is the address of a pointer.) All accesses to relocatable heap items must be dereferenced (followed through the handle to the pointer to the actual heap item) before a task can use the item. That

way, if an item is relocated by the heap munger (see description of the heap munger, later) during its background heap-optimization, any tasks that own the relocated item will still be able to find it because the heap munger always fixes the master handle so it is correct after moving a heap item. This master handle is often located in the master handle array, although it doesn't need to be. Wherever it is, though, it must be in a nonrelocatable place so that the heap munger can follow a pointer back

from the heap item to its master handle.

After the last master pointer is an end mark. At the next 32-byte boundary, the heap begins.

The heap (Figure 3, page 47) takes up all the available RAM beyond the system zone. It is a single data structure composed of chunks called items. Every single byte of the heap belongs to an item of one sort or another. An item can be a deletion, or it can contain actual information. Items that contain information can be allocated (owned by one or more tasks) or unallocated. Unallocated items can be purged by the heap munger (see later) if it needs to make more room for a memory request from a task. For speed and ease of programming, every single heap item begins and ends on a 32-byte boundary.

Virtually all system information that doesn't have to be addressed absolutely is stored in the heap, in various heap items owned by the system kernel. In addition, the heap contains all executable code, including the system kernel itself, in chunks called code items.

Each heap item contains a 32-byte heap header record, followed by zero or more 32-byte data blocks. The header record contains the information necessary for the heap manager and the heap munger to identify, move, and validate each item. The heap is organized, like much of the rest of the system, as a doubly linked list. As the heap munger scans through it, it follows the pointers forward or backward to verify that all is in order. If it finds anything that is not completely kosher, it immediately stops the system, takes over the system console, and enters the debugger with a descriptive error message. When a bug occurs, it is often the heap munger that detects the problem (in the form of a messed-up heap header) before anything else happens.

Division of Labor

The kernel is divided into several distinct blocks of code. They are of three types: one-time routines (initialization), system calls (routines available from every task), and discrete tasks (self-contained programs that run under the context switcher, just as do application tasks). Some of the system

It's 3AM!



Do you know where your bugs are?

This C programmer is finding his bugs the hard way...one at a time.
That's why it's taking so long. But there's an easier way. Use

PC-Lint

PC-Lint* analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. PC-Lint resembles the Lint that runs on the UNIX* O.S., but with more features and some awareness of the 8086 environment.

- Full K&R C
- Supports Multiple Modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- User-modifiable library description files for most major compilers.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or prior to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Price: \$139.00 MC, VISA**
(Includes shipping and handling) PA residents add 6% sales tax. Outside USA add \$10.00
- Runs under MS-DOS* 2.0 and up, with a minimum of 128Kb of memory. It will use all the memory available.

Trademarks: PC-Lint (Gimpel Software), UNIX AT&T, MS-DOS (Microsoft).

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

Call name	Description
(Task Manager Calls)	
Spawn	Create a new task
Kill	Destroy a task (by task number)
Suicide	Kill the calling task
(Heap Manager Calls)	
HeapGimme	Allocate an item in the heap
HeapDel	Release (delete) a heap item
FillZero	Re-initialize a heap item
GetMaster	Assign a master handle for an item
(Message Manager Calls)	
SendMsg	Send a copy of a block of memory to another task
Del1Msg	Delete message from top of incoming queue
DelMsgs	Delete entire incoming message queue
TxtMsg	Send a message of type "TEXT"
GetMsg	Fetch next message in queue
HandleMsg	Analyze incoming message and handle if standard type
(Character I/O Manager Calls)	
DevReq	Request a character I/O channel
DevDemand	Demand a character I/O channel (usually impolite)
DevRel	Release a character I/O channel
PrToStd	Select this task's standard character I/O device
PrToMem	Select the MemPrt device (see text)
(Text Manager Calls)	
GetCommand	Input a command line and parse it, passing control to the appropriate routine based on the command.
AddCmdTab	Add a set of commands to the existing command set
DoCommand	Parse and execute a command already stored in memory
GetPSW	Input and encrypt a password (to be compared with an encrypted password from the user file)
MoveString	Move an ASCII string
GetLine	Input a line of ASCII text from the character I/O device
PrLine	Print an ASCII string on the character I/O device
Print	Print a line of text. The line is expected to immediately follow the JSR Print instruction.
CompString	Compare two ASCII strings
(Miscellaneous System Calls)	
Random	What system would be complete without random numbers?
Sqrt	Square root of 32-bit integer

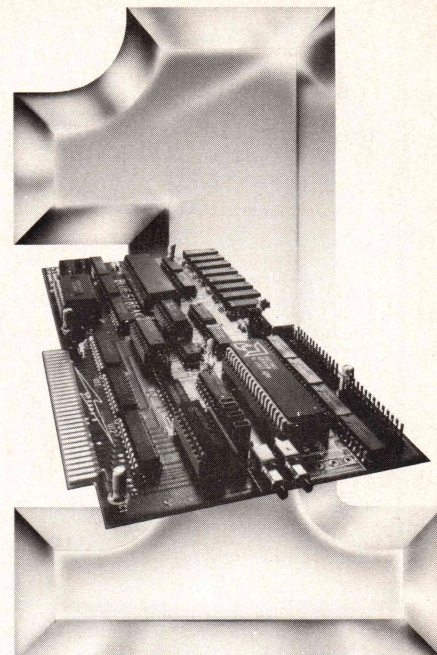
Table 1: Some of Terra Nova's system calls

TRAP-oriented calls			
Instruction		Cycles used	Description
TRAP	#n	38	call the system routine
MOVE.L	2(SP),A0	16	point to word argument
MOVE.W	(A0)+,D0	8	fetch the argument
MOVE.L	A0,2(SP)	16	update return address
(variable)	(variable)		decode argument word
-----			useful code
RTE		24	return to caller

		104 (+ decode)	total cycles for overhead
JSR-oriented calls			
Instruction		Cycles used	Description
JSR	Label.W	18	call low memory entry point
JMP	Label.L	12	call actual routine
-----			useful code
RTS		16	return to caller

		46	total cycles for overhead

Table 2: Comparison of TRAP- and JSR-oriented system calls



**Number One
in Performance**

**Z80H
BLUESTREAK™**

IBM/AT/XT/PC- 8mz

No Wait States

FEATURES

- 64K-256K RAM
- 2K-8K EPROM/Static Ram
- 2 Serial Ports
- Async/Sync/Bisync Communications
- Real Time Clock
- Memory-mapped Dual-port BUS
- On-board/Remote Reset NMI capability
- Up To 32 Boards Per AT/XT/PC
- Can Operate As Standalone Processor
- Less Than Full Size Board
(will fit other compatibles.)

SOFTWARE

- ZP/M™ CP/M Emulation Software
(Supports Most CP/M Software)
- Multiuser Capability If Used As A
Slave Processor

IBM is a registered trademark of International Business Machines.
CP/M 80 is a registered trademark of Digital Research Corp.



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle **no. 173** on reader service card.

9 TRACK TAPE CONTROLLERS AND 1/2" TAPE SUBSYSTEMS

MODEL TC-PC

TC-PC is a high performance 9-track tape controller for the IBM-PC with these important features:

- Reads and writes industry standard 1/2-inch tape
- Compatible with most formatted tape drives
- Standard 8-bit parallel recording with parity, and read-after-write verification
- Switch selectable I/O address (four contiguous ports required for operation)
- Maximum data transfer rate of 192,000 bytes per second
- Record length from 1 to 65,535 bytes
- Supports up to 8 tape transports
- Jumper selectable DMA channel
- Modes: PE and NRZI at 800, 1600, 3200 and 6250 bytes/inch
- Installable device drivers allow creation of application programs which run under IBM XENIX and MS-DOS
- Operates with IBM-PC and -XT; Compaq Portable; Zenith PC-150; Sperry PC; the Leading Edge Computer, and other 100% IBM-PC compatible equipment.

MODEL TC-50

TC-50 offers all the standard features of the TC-PC with these additional enhancements:

- Maximum data transfer rate of 400,000 bytes/second; 904,000 bytes/second with memory option
- Operation with a wider range of IBM-compatible machines, including IBM-AT; Compaq Desk Pro; ATT 6300 and others

A variety of software utilities is supplied as part of the TC-PC and TC-50 packages, including:

- **DEPOT (Data Exchange Program with Optional Translation)**
DEPOT provides a means to transfer data between system disk and magnetic tape, allowing:
 - Data interchange from tape to disk, and disk to tape
 - Conversion from ASCII to EBCDIC, and vice versa
 - Positioning to arbitrary location prior to data read
 - Specification of record length and block factor when writing from disk to tape; allows deblocking when reading from tape to disk
 - Multiple operations to be specified from a command file
- **TAU (Tape Archive Utility)**
 - Provides individual file backup and restore
 - Allows use of MS-DOS wild cards such as "*" and "?"
 - Provides disk drive selections for I/O
 - Changes pathname selections from within TAU
 - Provides data encryption for security

WARRANTY

All Overland Data products carry a 30-day unconditional money-back guarantee, and are warranted for one year, parts and labor.

OVERLAND DATA, INC.

5644 Kearny Mesa Road #A
San Diego, CA 92111
Tel. (619) 571-5555

XENIX and MS-DOS are Registered Trademarks of Microsoft Corp.
IBM PC/XT/AT are Registered Trademarks of International Business Machines Corp.
Compaq Portable and Compaq Deskpro are Registered Trademarks of Compaq Corp.
ATT 6300 - AT&T Information Systems Corp.
Sperry PC - Sperry/Univac Corp.
Zenith PC-150 - Zenith Data Systems
Leading Edge is a Registered Trademark of Leading Edge Products, Inc.

Circle no. 192 on reader service card.

PCT_EX

Now for PC Users:

Professional Typesetting Capability

PCT_EX brings to the personal computer user the ability to put any kind of information on paper in a professional, elegant manner. It brings the full power and flexibility of T_EX implementations on mainframes to owners of IBM PC's, AT's and workalike computers.

PCT_EX is widely used for formatting technical and mathematical material. It is also perfectly suited for producing professional-quality reports, manuals, even books.

PCT_EX offers a wide range of typefaces, and a wide choice of drivers which output the finished material on dot matrix printers (Epson, Toshiba), low-cost laser printers (Apple LaserWriter, Corona LP-300, HP Laser Jet) and graphics screen preview (Hercules, EGA). This ad was formatted by PCT_EX and produced on a Corona LP-300.

Join hundreds of satisfied PCT_EX users. Write or call us today.

PCT_EX: only \$279. Dot-matrix drivers: \$100. Laser drivers: \$300. Preview (Hercules GC): \$250. MF Medley (44 fonts, including Computer Helvetica): \$100. Corona Laser Printer and PCT_EX: complete \$3395. System requirements: DOS 2.0 or later, 512K RAM, 10M hard disk. M/C, Visa accepted.

Personal
T_EX
Inc

20 Sunnyside, Suite H
Mill Valley, CA 94941
(415) 388-8853 Telex 275611

Trademarks: PCT_EX, Personal T_EX, Inc.; T_EX, American Mathematical Society; IBM PC and AT, IBM Corp; LaserWriter, Apple Computer, Inc.; Hercules Graphics Card, Hercules Computer Technology.

Circle no. 76 on reader service card.

MULTITASKING OS
(Continued from page 48)

calls we've developed are listed in Table 1 (page 49). This is not intended to be a complete list, only to give some of the system's flavor.

The initialization section is the block of code that gains control before anything else happens. It starts with a brute-force approach: It grabs control from whatever operating system invokes it, and then it sets up the bare-bones data structures for the system. I'll go into greater detail about the initialization section later, when I talk about tricks and shortcuts.

The most important low-level code segment is the context switcher, which is the routine that receives control from one task and passes it on to the next. It is extremely small and extremely fast, and it makes a lot of assumptions about the tasks as it does its job. This is by design: By making assumptions and forcing the tasks to adhere to them, a lot of overhead is eliminated. Again, I'll go into detail about the context switcher in the section on tricks and shortcuts.

The task manager is composed of a group of system routines available to every task. They allow a task to create, destroy, and manipulate other tasks or itself.

The heap manager is a collection of routines that allow any task to request, release, lock, enlarge, or otherwise manipulate heap items.

The message manager is a collection of system routines that make possible a clean and well-defined message-passing protocol between tasks. Simply by pointing at a block of memory and calling a system routine, any task can send a copy of any piece of data to any other task. Reading queued messages from other tasks is similarly easy.

The character I/O manager is a set of system calls and interrupt service routines. Together with the text manager, it makes possible a simple I/O structure, in which each task can select any physical or logical device for I/O by passing a device number. Serial input is interrupt-driven, and serial output is polled. Device drivers can be added or removed from the I/O manager with another set of system calls.

The text manager is a collection of

system routines that ease the processes involved in talking to humans. It includes powerful routines to get and parse command strings as well as text-manipulation routines such as case conversion, context-sensitive string comparisons, and so forth. The powerful parsing calls make it easy to create a tiny machine-language task that includes a complete command interpreter and syntax error handler. This is very important if a significant amount of development is to be done in assembly language.

The trap manager handles all system traps except I/O interrupts, which go directly to the I/O manager. Error traps always cause the system to come to a complete standstill. This is important to us because of the close interaction between the tasks, which must always be in intimate communication to fulfill the purpose of the system. When an error trap occurs, all task switching and interrupt processing stop and the task in which the error occurred takes over the system console and enters the debugger. The human in charge can then take corrective action and restart the system with minimal damage. Obviously this approach would be completely unacceptable in a commercial operating system, but for us, it is ideal because we have all the source code for the entire system and can often correct bugs as soon as they occur.

No assembly language development system is complete without a debugger, of course, especially a multitasking one. Our debugger is a command parser that any task can invoke, either automatically (in response to an error trap) or directly. It is capable of running alongside other active tasks, even multiple copies of itself, and it allows the user full manipulation and examination of memory.

The heap munger is a distinct task, always present, always active, whose original job was to survey the contents of the heap continuously and maintain it as an efficient data structure (by using a background task for this, we avoided many complexities). The heap munger has turned into quite a bit more than just a trash compactor, however. Its responsibilities are many and varied, from checking the TCB array for in-

tegrity and waking up sleeping tasks when their ships come in to responding to messages from other tasks that want to know what the system loading is so that they can adjust their own CPU usage to increase the overall performance of the system. In fact, the heap munger is also capable of responding to text messages sent to it by a human who is operating a user task, in which case it responds by sending a plain English message back to the source task, which then displays it for the human to read. As a general rule, the heap munger performs any systemwide activity that must be performed at frequent, regular intervals. With all these responsibilities, it is the largest single code

segment in our kernel, weighing in at about \$A00 bytes.

The disk munger, like the heap munger, is a distinct task that is always running except when it's waiting for an I/O completion. Because its structure and function are application specific, I won't go into it in great detail here. I would like to point out, however, that by allocating a single task to handle each disk device, a large number of problems related to data contention between tasks can be avoided. The disk munger is entirely message-driven: As each task requires a disk access, it sends a message to the disk munger for the device it wants to access. Each I/O request gets added to a queue of such

Oh, Rapture!

This is truly the editor I have been longing for.

- Dr. Joseph Newcomer, Software Engineering Institute

New Epsilon 3.0: fast, *fully programmable* text editor with an EMACS-style command set and concurrent processes!

Presenting Epsilon 3.0, the fastest, most powerful text editor available for personal computers. Epsilon has a built-in programming language, called EEL, for creating your own commands. Plus you get EEL source code for all of Epsilon's commands!

EEL has all the expressive power of the C programming language. It supports all C statements and expressions, pointers and user-defined structures. Unparalleled flexibility!

Because EEL looks like C, commands are easy to write. You don't have to learn a new language. Epsilon detects illegal pointer references, and has a source line single-stepping debugger and EEL profiler, too.

Our amazing Concurrent Process facility lets you run other programs while you continue to edit your files. The program's input and output are connected to a window, so you can edit them. Great for background compiles, debugging while looking at source code, and lots more!

Plus the advanced features programmers need:

- | | |
|--------------------------------|------------------------------------|
| - Concurrent Processes | - Context Sensitive Help |
| - Multiple Windows | - Regular Expression Search |
| - Unlimited File Size | - Unlimited Number of Files |
| - Customizable Keyboard | - File Name Completion |
| - Tutorial | - Convenient Keyboard Macros |
| - Automatic Swap File | - Directory Perusal |
| - Supports Large Displays | - Language Support (C, Lisp, etc.) |
| - Saves Deleted Text (n times) | - Uses All Available Memory |

Epsilon runs on IBM PC's, XT's, AT's and compatibles with PC-DOS 2.0 or above and requires 256K of memory.

Epsilon is available directly from Lugaru Software Ltd, and costs only \$195.00. Order now using your Visa, MasterCard, or American Express card. Company purchase orders are also welcome. Order it today, so you can enjoy it soon!

Lugaru Software Ltd.
5740 Darlington Road
Pittsburgh, PA 15217
(412) 421-5911

Circle no. 135 on reader service card.

C68 & C68/020 C COMPILERS for MC680X0

- ▶ Produce highly optimized code
- ▶ Complete development environment: Assembler, Linking and Downline Loaders, Runtime Libraries
- ▶ Available for Motorola, DEC, and Alcyon host computers
- ▶ The #1 choice for compact, fast MC680X0 code
- ▶ \$1495 for C68 (Motorola host)
\$2295 for C68/020 (Motorola host)



5010 Shoreham Place
San Diego, CA 92122
(619) 587-1155 TELEX 5106004947

DEC is a Trademark of Digital Equipment Corporation.

Circle no. 221 on reader service card.

**THE BEST Z80
ASSEMBLER ON
THE MARKET JUST
GOT BETTER!**

Z80ASM
NOW ONLY **\$49.95**

**DON'T ASK HOW OURS CAN BE SO FAST ...
ASK WHY THEIRS ARE SO SLOW!**

"... a breath of fresh air ..."

Computer Language, Feb. 85

"... in two words, I'd say speed & flexibility",

Edward Joyce, User's Guide #15

Now fully compatible with M80 in .Z80 mode with many extensions. Time & date in listing, 16 char. externals, plus many other features.

To order, or to find out more about our complete family of development tools, call or write:

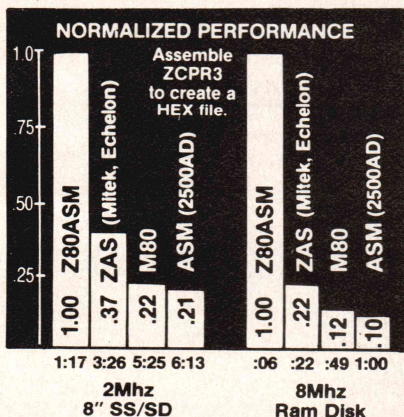
SLR Systems

1622 N. Main St., Butler, PA 16001
(800) 833-3061, (412) 282-0864
Telex 559215 SLR SYS



C.O.D., Check or Money Order Accepted

SHIPPING: USA/CANADA + \$3 • OTHER AREAS + \$10
Z80 CP/M compatibility required.



Circle no. 78 on reader service card.

MULTITASKING OS
(Continued from page 51)

messages. When the disk I/O is completed, the disk munger sets the "wake-up" flag for the requesting task, and the heap munger wakes it up on its next pass. The requesting task then looks in its TCB for the completion code from the disk munger. Because the I/O requests are high-level calls, usually implicitly including the open, read/write, and close of the file, there are few file contention problems.

On start-up, a discrete task called the system console manager initially gains control of the system console and puts up a system monitor prompt. Until another user task is spawned, it is the only task that has access to a character I/O device (and thus, to a human). The system console manager can then give way to any other user program.

Tasks

Everything that gets done on the system, with the single exception of the context switch between tasks, is done by a task. Three tasks are always present in the system: the system console task, which initially runs the system console manager; the heap munger; and at least one disk munger. The disk munger is actually optional, although it's hard for me to imagine doing any useful work without using a disk.

Each task, whether it's active or not, possesses exactly one task data item (TData item) in the heap. The TData item is crucially important to the proper functioning of the system because it's where each task stores its most important local information. Every task's TCB entry contains the master pointer to its TData area in the heap. During normal execution of a task, the 68000 register A5 always points to the base of the TData item. Before the context switcher passes control to a task, it always sets up register A5 for the incoming task.

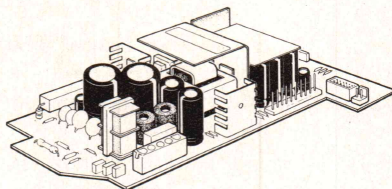
The TData item is also the location of the task's local data stack: The stack goes from the top of the TData item down, and the task's local data goes up from the bottom. Note that it is the responsibility of each task to ensure that its stack and TData areas do not collide.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

- + 5VDC - 8 Amps
- +12VDC - 5 Amps
- 12VDC - 1 Amp
- (81 WATTS MAX)



\$29⁹⁵
ea.

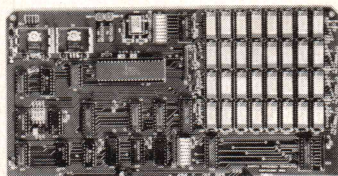
4 FOR \$99

ADD \$2
EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

256K S-100 SOLID STATE DISK SIMULATOR! WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB
(WITH CP/M* 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$69⁹⁵
(8203-1 INTEL \$29.95)

(ADD \$50 FOR A&T)

\$149⁰⁰

#LS-100

(FULL 256K KIT)

- FEATURES:
- * 256K on board, using + 5V 64K DRAMS.
 - * Uses new Intel 8203-1 LSI Memory Controller.
 - * Requires only 4 Dip Switch Selectable I/O Ports.
 - * Runs on 8080 or Z80 S100 machines.
 - * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
 - * Provisions for Battery back-up.
 - * Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
 - * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
 - * Compare our price! You could pay up to 3 times as much for similar boards.

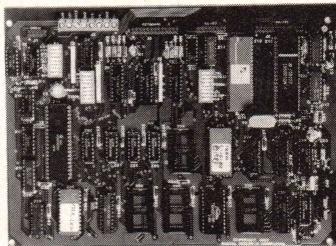
THE NEW ZRT-80

CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- * RS232 at 16 BAUD Rates from 75 to 19,200.
- * 24 x 80 standard format (60 Hz).
- * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- * Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- * Composite or Split Video.
- * Any polarity of video or sync.
- * Inverse Video Capability.
- * Small Size: 6.5 x 9 inches.
- * Upper & lower case with descenders.
- * 7 x 9 Character Matrix.
- * Requires Par. ASCII keyboard.



\$89⁹⁵

#ZRT-80

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716
CHAR. ROM. 2732 MON. ROM

\$49⁹⁵

SOURCE DISKETTE - ADD \$10
SET OF 2 CRYSTALS - ADD \$7.50

FOR 8 IN. SOURCE DISK
(CP/M COMPATIBLE)
ADD \$10

64K \$100 STATIC RAM

\$119⁰⁰
KIT

LOW POWER!
150 NS ADD \$10

BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

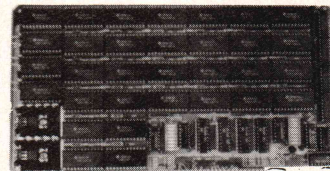
SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 S100
STANDARD
(AS PROPOSED)

FOR 56K KIT \$105

ASSEMBLED AND
TESTED ADD \$50



FEATURES: PRICE CUT!

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

PANASONIC

Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

\$29⁹⁵

EA.

GROUP SPECIAL:

4 for \$99⁰⁰

WITH DATA & SCHEMATIC

(USA SHIPPING: \$3. PER UNIT. CANADA: \$7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

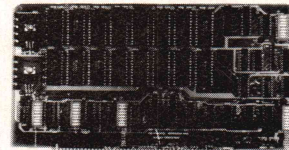
32K S100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II
FULL
EPROM KIT
\$69.95
A&T EPROM
ADD \$35.00



BLANK
PC BOARD
WITH DATA
\$39.95

SUPPORT
ICs
PLUS CAPS
\$16

FULL
SOCKET SET
\$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- * This one board can be used in any one of four ways:
 - A. As a 32K 2716 EPROM Board
 - B. As a 32K 2732 EPROM Board (Using Every Other Socket)
 - C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
 - D. As a 32K Static RAM Board
- * Uses New 2K x 8 (TMM2016 or HM6116); RAM's
- * Fully Supports IEEE 696 Buss Standard (As Proposed)
- * Supports 24 Bit Extended Addressing
- * 200 NS (FAST!) RAM's are standard on the RAM Kit
- * Supports both Cromemco and North Star Bank Select
- * Supports Phantom
- * On Board wait State Generator
- * Every 2K Block may be disabled
- * Addressed as two separate 16K Blocks on any 64K Boundary
- * Perfect for MP/M* Systems
- * RAM Kit is very low power (300 MA typical)

32K STATIC RAM KIT — \$99.95

For RAM Kit A&T — Add \$40

TERMS: Add \$3.00 postage. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

Each task's TData item contains spaces for various pointers and vectors associated with its current character I/O device, if any. The vectors include various standardized routines such as InCheck, which checks to see if a character is available; InWait, which waits for a character and returns with it; OutCheck; OutWait; plus several others. The pointers include the addresses of the destination for the next input byte (if any), the source of the next output byte, and so forth.

Character input is generally interrupt-driven. Because a task that is awaiting input is often "asleep" (not in the active TCB list), it is necessary for the interrupt routine to set a flag that tells the system to wake up the owner of the device. Then, after the interrupt has been handled, the heap mungers, which is always awake and active, catches the set flag the next time it gets control and performs the actual manipulations to return the task's TCB to the active list. In order to

provide an input time-out, the heap mungers also awakens each task once every ten seconds so that the task can test for the time-out.

Each task may request to be assigned to a character I/O device. If a device that is requested is currently assigned to another task, the requestor will usually have to wait until the device is available. For special cases such as error traps, however, there is a system call that allows the task to demand to be connected to a device. When a demanded device is released, it reverts to the task (if any) to which it was attached originally. When a requested device is released, it always becomes available (unattached) again. Some devices can be attached to multiple tasks at the same time—for example, there is a device called MemPrt that reads and writes to memory as if it were a character stream from/to a serial device. This "virtual" device can be simultaneously active for different tasks, each with its own set of pointers in the TData item for reading and writing.

I won't go into great detail about our I/O drivers or the low-level struc-

ture of the I/O routines. This sort of information is readily available in several forms in computer bookstores.

The Good Stuff: Tricks and Shortcuts

The first thing our operating system does when it gains control of the processor is to deviously remove the existing operating system. This it does by using a trick to get into supervisor mode (see the listing, page 102). First, it shoves a new address into the privilege exception vector, which is in the hardware vector table in low memory. This address happens to be that of the second instruction following the one that does the store to the vector. The next instruction is a privileged but otherwise harmless one. Thus, when it tries to execute it, it traps to the privilege exception vector and thence to the next instruction in the program. If through some quirk we happen to be in privileged mode already, the processor harmlessly executes the privileged instruction and falls through. Now we are in privileged mode, and we can quickly grab the rest of the system.

Next, we turn off all the interrupts in the system as quickly as possible. It is important to do this before clearing memory because a stray interrupt might happen before we can turn it off and it must still vector properly. After all interrupts are out of commission, we copy our own set of vectors into the interrupt table.

The jump table (discussed later) is next moved into place in low memory. It is read from a section of object code within another assembled module. It's nothing more than a sequence of more than a hundred absolute-long JMP instructions.

Next, we clear the rest of memory—except the kernel, of course—to zeroes. This is both a general safety measure and an aid in debugging: If a chunk of memory is nonzero, we can be certain that something we did caused it to be that way. Also, it's nice to be able to assume that unused memory is always zeroed out; it makes for much faster initializations later on, once the system is running.

The system zone requires a certain amount of initialization. The task control blocks must be set up and the end marks for the TCB and master

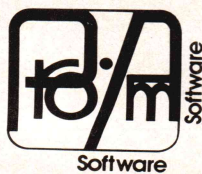


DISnDATA, The Only Disassembler That Tracks Down DATA!!!

- Fully disassembles both .EXE and .COM files!
- Performs recursive flow- and Segment Register data-trace to determine SEGMENT, PROC & Data Areas (even within 'CODE' segments!)
- Outputs appropriate SEGMENT and PROC pseudo-ops at proper places within the assembly-language output!
- Outputs data areas using most appropriate form of DB or DW (ASCII printable text as a character string, others as their hex value).
- Chooses data lengths (DB or DW) to match byte or word data references in code, allowing most memory references to be free of BYTE or WORD length operators.
- Outputs large, all-zero areas with "DB/DW nn DUP (?)" to prevent excessive output from large buffers, uninitialized arrays, etc.
- Fully labels both code and data. Labels are of the form 'Hxxxxx', where 'xxxxx' is the hex offset of labelled item from the beginning of the program.
- Outputs code, data & pseudo-ops in IBM* ASM or MASM assembler format. (Output may be directed to display, printer, and/or disk.)
- For IBM* PC*/XT* & compatibles, 128K+ RAM, 1 or more disks, DOS 2.x.

#8634-20 PC-DISnDATA 1.0 (SSDD 5-1/4" diskette) and manual \$145

U.S. Funds Only. Add \$3 shipping (U.S. & Canada), \$10 (overseas air) per item. Ohio residents please add 6% sales tax. *Registered trademark, IBM Corporation



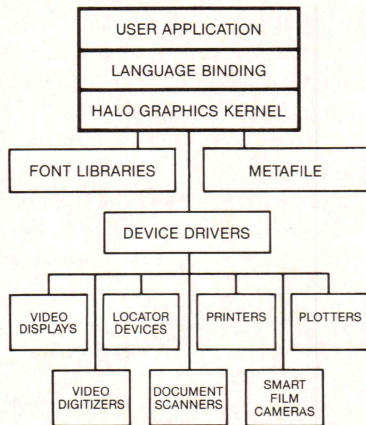
To order, phone (513) 435-4480 (M-F, 9 A.M.-5 P.M. EST), or send check, money order, or VISA/MasterCard information (name, street address (no P.O. Box please), card number, expiration date, and your telephone number) to:

PRO/AM SOFTWARE
220 Cardigan Road
Centerville, OH 45459

Professional Software for
both Novice and Expert

HALO the market standard for graphics

HALO provides software implementors with a complete graphics development environment.



Device Independent

Each copy of HALO comes with a full complement of display, input, printer, film recorder and plotter device drivers. Metafile creation and display are included. Media Cybernetics is continually developing new device drivers to meet the ever changing new hardware technologies.

HALO supports all IBM compatible MS/PC DOS systems including Texas Instruments Professional™, AT&T 6300™, IBM™ 3270-PC and of course the IBM PC, XT and AT. Over twenty graphics display boards manufactured by IBM and leading third party vendors are supported. A single device driver supports all display modes on each device. Image input from video digitizers and document scanners is supported. New device drivers include the IBM EGA and Professional and the AT&T Image Capture and Display boards.



This picture was created with "Artwork", a HALO-based application written by West End Film of Washington, DC.

De-Facto Standard

First released in 1982, HALO has an established base of over 40,000 end users and over 100 corporate clients. Numerous HALO-based CAD, solids modeling, presentation graphics, art packages, mapping and other vertical market software applications are commercially available.

As early as June 1984, PC World featured an article entitled, "HALO A new software library leads the way toward graphics stand-

ardization and portability." The July 9, 1985 issue of PC Week™ featured users stories from both Rockwell International Corp., and Lawrence Livermore Labs on how they use HALO to save development time and money. Most recently, Mini-Micro Systems August 1985 issue stated, "Widely used, HALO has attained de facto-standard status. It's certainly the most widely used library." HALO has achieved this status because it provides a complete device independent graphics environment for software developers. Since the HALO interface rarely changes, compatibility with a new device is achieved simply by adding a new device driver.

By incorporating all the HALO device drivers into their applications, licensed commercial software developers solve distribution and compatibility problems.



This picture was created with "Design Board 3-D", a HALO-based application written by MEGA-CADD of Seattle, Washington.

Powerful Graphics Toolbox

HALO has over 170 functions. Standard functions include: point, line, arc, circle, ellipse, polygon, etc. Coloring control functions are available for color selection, palette management, dithering, and textures.

Advanced features encompass: image compression routines for reducing data storage requirements; both absolute and relative addressing plus individual cursors for graphics text and interaction; world and normalized coordinate systems for windowing, viewport and clipping; and a metafile reader and generator.

HALO's raster extension supports rubberbanding and non-destructive modes so that polygons and text may be interactively created and moved over a displayed image.

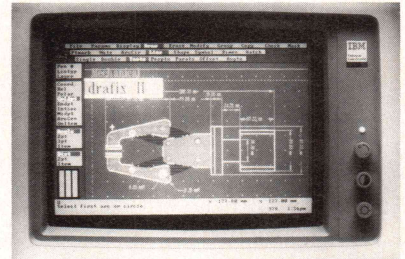
There are currently over twenty bit mapped and stroke text fonts available for HALO. The object oriented type faces provide user definition of line width, size, proportional width, angle, and interior fill style.

HALO's Virtual Rasterization Interface (VRI) frees users from the limited resolution of their graphics board by modeling the graphics display in memory and having the hardcopy produced in the maximum resolution of the output device.

HALO can be addressed as a linkable library or as a resident driver depending on the application.

Faster than VDI

In benchmark tests by independent companies evaluating HALO and GKS, HALO was determined to be three to ten times faster depending on the function tested. These benchmark results made HALO a requirement for many government agencies and large corporations. This gives HALO compatible hardware and software significant inroads into these agencies and corporations.



This picture was created with "drafix II", a HALO-based application written by FORESIGHT Resources Corp. of Overland Park, Kansas.

Easy to Learn

HALO comes with a computer aided tutorial. This tutorial, entitled LEARNHALO, teaches users to become proficient graphics developers in less than a day by providing step-by-step instructions through graded examples.

HALO is written in Assembler and callable from the language you are already comfortable with: Interpretive and Compiled Basic, Professional Fortran, MS Fortran™, Pascal, Turbo Pascal™, MS C™, Mark Williams C™, Lattice C™, Aztec C™, Computer Innovations C86™, and Golden Common Lisp™.

Increase Market Value of Hardware

Graphics hardware manufacturers receive distinct marketing advantages by having their products supported by HALO. HALO is the gateway to an extensive body of commercially available software.

Pricing Information

A single copy of HALO for one language costs \$250.00. A second language binding ordered at the same time is an additional \$150.00. Author, Corporate and Site licensing and OEM agreements are negotiated on a case by case basis.

To order, or receive more information on HALO and other Media Cybernetics, Inc., products contact:

media cybernetics, inc.

7050 Carroll Avenue
Takoma Park, MD 20912
(800) 446-HALO or (301) 270-0240
Telex #332014

HALO, LEARNHALO, Texas Instruments Professional, AT&T, IBM, MS Fortran, Microsoft C, Turbo Pascal, Mark Williams C, Lattice C, Aztec C, Computer Innovations C86, Golden Common Lisp and GKS VDI are all registered trademarks of Media Cybernetics Inc., International Business Machines, Texas Instruments Inc., AT&T, Microsoft Corp., Borland Intl., Mark Williams Co., Lattice Inc., Manx Software Systems, Computer Innovations Inc., Gold Hill Computers, and Graphics Software Systems Inc., respectively.

MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree ISAM DOES THEM ALL!

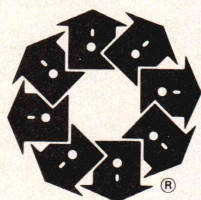
The creator of Access Manager™ brings you the most powerful C source code, B+ Tree file handler: **c-tree™**

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

\$395 COMPLETE

Specify diskette format:

- 5¼" MS-DOS
- 8" CP/M
- 3½" Mac
- 8" RT-II



For VISA, MC and COD orders
call (314) 445-6833

FairCom

2606 Johnson Drive
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle no. 93 on reader service card.

Scrap your LINKER with FASTER C

Reliably:

CUT Compile times (by 15% to 55%)

CUT Testing times (by 12% to 37%)

HOW: FASTER C keeps the Lattice C or C86 library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

"Automatic" support for new libraries by reading the .OBJ files makes support for new libraries quick and simple.

AVAILABLE FOR PC-DOS, IBM-AT,
AND ANY 256K MSDOS SYSTEM.

ONLY \$95.

Full Refund if not satisfied

during first 30 days.

Call 800-821-2492

**Solution
Systems™**

335-D Washington St., Norwell, Mass. 02061
617-659-1571

Circle no. 155 on reader service card.

MULTITASKING OS

(Continued from page 54)

handle arrays must be set in place. The values in the miscellaneous system data area must also be initialized.

After the system zone is in place, the heap is defined from the next 32-byte boundary to the end of memory. Three heap items are initially set aside: a deletion from the beginning of the heap to the start of the kernel's code, a fixed (immovable) code item for the kernel, and another deletion from the end of the kernel to the end of memory. Soon, the other tasks will be carving up the big deletions for their own use.

Each time a task is spawned, the task manager creates a new TCB and a new TData item. The first task to be spawned is the system console manager. The system task manager allocates an item of the proper size from the heap, creates a TCB for the new task, and adds the TCB to the current linked list of active tasks' TCBs.

After the system console task is spawned, the heap munger and disk munger are also spawned. Note that none of them begins to execute until the initialization code jumps into the middle of the context switcher.

We discovered that one of the biggest sources of "extra" system overhead in commercial operating systems is the need to manage tasks that might possibly get out of hand and take over the system. In most cases, such "runaway" tasks are avoided by using a hardware timer to assure that any given task will only be able to run for a preset time. If a task spends too much time without releasing to the system, the timer interrupt occurs and vectors the CPU through to the supervisor program. This program then saves the existing task's registers and status and restores those of the next task in line, which then is off and running with a newly reset timer. All of this requires a lot of tricky and complex code to ensure that no task can "run away" and lock up the system. Also, the constant saving and restoring of registers and status, which is necessary because the context switch is interrupt-driven, adds measurably to the overhead required for a context switch.

We debated for some time about the best way to reduce this overhead.

Finally, we settled on the answer: We developed our kernel as a non-preemptive task controller. This means that there is no hardware timer to interrupt each task after some preset interval, and no routines are required to service such an interrupt. Instead, we use a simple context switcher, one that doesn't even bother to save registers or the previous task's status bits. It is the task's responsibility to call the context switcher often enough to ensure smooth system operation and to make sure that it saves any registers that it needs (other than A5 and the stack pointer). For our application, this is perfect because most of our tasks spend most of their time waiting for input or output in the inactive task list, during which time the other tasks can run unhindered.

When a task has finished with the CPU and is ready to let the next task in the active list run for a while, it simply calls the context switcher as a subroutine using a JSR instruction to a low-memory JMP instruction whose address is fixed regardless of the location of the context switcher. Even with the added overhead of the extra JMP instruction, this method of calling is considerably faster than a TRAP instruction—the usual method of calling a context switcher.

Once the switcher has control, it checks to see if the system tasking is stopped (see the listing). If it is, then the calling task immediately gets back control through a simple RTS. Otherwise, it gets ready to call the next task.

The address of a task's TCB is in its TData area. This address is loaded into A0. Then the current TData base address (in register A5) is subtracted from the stack pointer, yielding a relative displacement, which is stored in the TCB. Now we're ready to move on to the next task.

The address of the next task in the circular linked list of active tasks is fetched from the old task's TCB into A0. Register A5 is set to point to the new task's TData area by moving its address from the TCB, and the stack pointer is restored from the relative displacement by adding A5 to it. Then a simple RTS returns control to the task.

This otherwise trivial scheme has one slight complication: Frequently,

a task will release control with the intention of going to sleep for a while. This happens, for example, when the RAM buffer has no input characters for the character device attached to a task that is waiting for input. When the input finally happens, the interrupt routine sets a flag that causes the heap manager to wake up the task when it next checks for such a situation. The problem is that the context switcher I have just described has no provision for putting the old task to sleep: It assumes that both tasks want to stay awake.

So, we created an alternate context switcher that removes the outgoing

task from the active list before calling the new one (see the listing). When a task wishes to go to sleep, it simply calls the alternate context switcher. The primary difference is that before it moves on to the next task, the alternate switcher does some standard and fast list manipulation.

Most commercial operating systems use one or more of the TRAP instructions to perform system calls, usually going on the premise that they are there for that purpose and that the TRAP instructions allow programs to be more general and more easily relocated. Unfortunately, TRAP instructions on the 68000 cause a ma-

Lattice® Works

NEW PROGRAMMER'S SCREEN EDITOR INTRODUCED

Designed specifically for programmers, the Lattice Screen Editor (LSE) is a fast and flexible, multi-window editor that is also easy to learn and use.

LSE runs under MS-DOS or PC-DOS on most popular machines with 128Kb memory. It provides standard editor functions such as block moves, pattern searches, and "cut and paste". In addition, LSE offers special features for programmers such as an error tracking mode and three assembly language input modes.

A complete installation program is included to remap any of LSE's 48 keyboard functions. Menus, prompts, help messages and default file extensions can also be customized for individual user preferences. \$125.00.

LATTICE TOPVIEW TOOLBASKET NOW AVAILABLE

Providing more than seventy functions, the Lattice TopView Toolbasket is designed for software developers writing applications for IBM's TopView multi-tasking, multi-window environment.



Lattice

Phone (312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES:

Benelux: De Vooght. Phone (32)-2-720-91-28. England: Roundhill. Phone (0672) 54675

Japan: Lifeboat Inc. Phone (03) 293-4711 France: SFDL. Phone (1) 666 1155

The Toolbasket functions eliminate the need for extensive use of assembly language when interfacing with TopView. The Toolbasket's library includes functions to control window, cursor, pointer, and printer operations. It also provides access to TopView's cut-and-paste facilities and offers debugging services.

The Toolbasket runs on the IBM PC, XT, AT, and compatible systems with 256Kb memory. \$250.00. Binary and source code is available for \$500.00.

LATTICE CREATES C COMPILER FOR COMMODORE AMIGA

Amiga C, produced by Lattice for the Commodore Amiga, supports the Amiga's 68000 microprocessor and offers the same high speed and extensive capabilities of the MS-DOS Lattice C compiler currently used by more than 30,000 software developers worldwide. Available from both Commodore and Lattice. \$300.00.

In addition, Lattice also offers cross compilers that allow you to develop Amiga programs on MS-DOS or UNIX systems.

Contact Lattice, to discuss your programming needs. Lattice provides C compilers and cross compilers for many environments including Tandy, Sony, Hewlett-Packard, Tandem, and IBM Mainframe. Corporate license agreements available.

Circle no. 101 on reader service card.

for problem: They take a long time to execute, as do the instructions to decode their arguments. The RTE instruction, which returns from a TRAP, shares the same problem.

We could understand the importance of relocatable programs, certainly, but we felt the long-lasting TRAP instruction was too much to ask of an ultra-fast, real-time system. We therefore designed a different, faster way to call system routines: the JSR

instruction and an old-fashioned jump table in low memory (see Table 2, page 49, for a timing comparison). Instead of using a TRAP instruction with an argument following in the next word, we simply call one of many entry points that are at absolute locations in low memory. Each entry point consists of a single JMP instruction to the actual system call entry point. Not only do we save the extra time required for the TRAP and RTE instructions but we also avoid having to extract the argument word from the bytes following the TRAP in-

struction and having to add two to the return address to jump around the argument because the argument is implicit in our choice of which routine to call. (With TRAP-based system calls, an argument is required to specify which system call to use because there are only 16 traps. With JSR calls, there can be hundreds of separate entries, so no argument is required to specify which call is to be used.) By using subroutines instead of traps, we shaved more than 100 machine cycles from every single system call, which makes a measurable difference in a machine that uses lots of system calls.

Conclusion

With the two context switchers just described, a small set of carefully designed system routines, a somewhat unusual system calling procedure, and a certain amount of cooperation from the application programs, we have vastly increased the throughput of our system. Our approach is obviously not well suited to most projects as it requires a considerable amount of skill and cooperation on the part of the programmers. Furthermore, because of its nonstandard nature, it is poorly suited to any applications that are written for commercial systems—at least until we get a C compiler running! If you need an extremely fast multitasking system for a specialized real-time application and are strapped for funds, however, this approach can turn a relatively inexpensive microcomputer into an amazingly powerful system. To date we have done just that for four different hardware configurations.

I'd be happy to discuss the philosophy and tricks in greater detail with anyone who is interested. For more information on our operating system, write to Nick Turner at the following address:

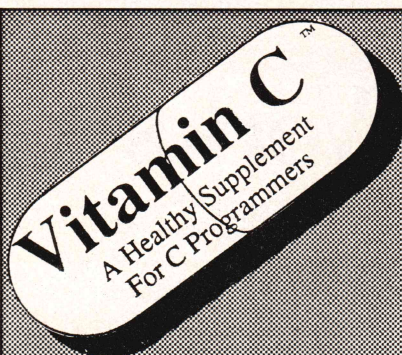
Terra Nova Communications
10 McGranahan Ct.
Boulder Creek, CA 95006

Or call me at (408) 338-9510. Terra Nova Communications is a consulting firm specializing in small multitasking systems. **DDJ**

(Listing begins on page 102)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**



100 % MONEY BACK GUARANTEE

Better than a brochure. More informative than a testimonial. Our guarantee gives you the opportunity to see first hand why programmers who demand performance are using Vitamin C.

Find out for yourself why Vitamin C users are saying...

The best structured and documented C source package we have ever seen.

Thanks to Vitamin C, our projects are back on schedule.

I own them all, but I USE Vitamin C!

Vitamin C! \$149.95 + \$3 ground, \$6 second day air, or \$20 next day. TX add 6 1/8% tax. Includes 100% source, reference manual, step by step tutorial, examples, sample programs. Specify Microsoft v3, Lattice, Aztec, Computer Innovations, DeSmet, or Mark Williams. Visa & MasterCard accepted. Ask about UNIX, TI-Pro, and other compatibility!

Data entry • Windowing Help System • More!

Finally! A library of high level C functions (not just a bunch of building blocks) designed to increase your productivity and help develop superior applications in dramatically less time! How? Well, Vitamin C automatically coordinates the complex tasks and leaves the programmer free to be creative! With Vitamin C, for example, you'll never even have to think about saving or restoring portions of the screen when a window is opened, closed or moved. Simply call `wopen()`, `wclose()` or `wmove()` and Vitamin C takes care of the complexities. It's just that easy! This philosophy of relieving the programmer from as many details as possible runs throughout Vitamin C. As a result, jobs that used to take days are finished in a matter of hours!

Vitamin C's powerful features include...

- Complete input formatting
- Unlimited validation
- Full attribute control
- Field sensitive help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- Date & time arithmetic routines
- "Loop function" allows processing while awaiting input

...and much much more!

---PLUS---

All windowing & data entry features are already fully integrated for effortless data entry windows, display windows, and pop-up menus!

Coming Soon... VCScreen!

Our new interactive screen "painter" actually lets you draw your input screens. Define fields, text, boxes & borders. Move them around. Change their attributes. Then after everything is just the way you want it, the touch of a button generates C source code calls to Vitamin C routines OR generates parameter files that will dynamically load & build each screen at run time. Either way, screen designs will be faster & more pleasing with VCScreen!

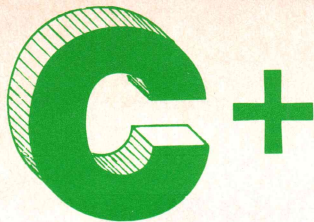
For Orders Or More Information...

(214)243-6197

Creative Programming Consultants

Box 112097

Carrollton, Texas 75011-2097



The Best C Book A Powerful C Compiler

One Great C Value **\$39.95**

A good C book just isn't complete without a good C compiler to go with it. That's why we give you both. You get a comprehensive 450 page book and a full feature standard K&R C compiler with the Unix V7 Extensions. The Book is loaded with examples that teach you how to program in C. And our fast one pass C compiler comes with an equally fast

linker so you don't waste a lot of time watching your disk drives spin. You also get a Unix compatible function library that contains more than 150 functions (C source code included). And if all that isn't enough, we offer you a 30 day money back guarantee. So what are you waiting for? The exciting world of C is just one free phone call away.

Language Features

- Data Types: char, short, int, unsigned, long, float, double
- Data Classes: auto, extern, static, register
- Typedef, Struct, Union, Bit Fields, Enumerations
- Structure Assignment, Passing/Returning Structures

abs
asm
asmx
atan
atoi
atol
bdos
bdosx
bios
biosx
calloc
ceil
cfree
chain
character
chdir
chmod
clearerr
close
clrscrn
cmpstr
conbuf
conc
cos
cpyst
creat
cursblk
curslin
curscol
cursrow
cursoff
curson
delete
drand
exec
execcl
execv
exit
exitmsg
exp
fabs
fclose
fdopen
feof
ferror
fflush
fgets
fopen
fputc
fread
free
freopen
fscanf
fseek
fstell
fwrite
getc
getch
putc
getchar
getcseg
getdseg
getd
putd
getdate
gettime
geti
puti
getkey
getmode
setmode
gets
getw
heapsiz
heaptrap
hypot
index
inp
insert
iofilter
isalnum
isalpha
iscntrl
isdigit
islower
isprint
ispunct
isspace
isupper
itoa
keypress
left
len
log
log10
longjmp
lseek
malloc
alloc
mathtrap
mid
mkdir
modf

Functions

movmem
open
outp
peek
perror
poke
poscurs
pow
printf
putc
putchar
puts
putw
rand
read
readattr
reach
writech
readdot
writedot
realloc
rename
replace
repemem
rewind
right
rindex
rmdir
scanf
setbuf
setbufsiz
setcolor
setdate
settime
setjmp
setmem
sin
sound
sprintf
sqrt
strand
sscanf
stacksiz
str
strcat
strcnv
strcpy
strlen
strncat
strncpy
strncmp
strsave
system
tolower
toupper
ungetc
ungetch
unlink
write
writechs
xmembeg
xmemend
xmemget
xmemput
xmovmem
_exit

MIX Editor \$29.95

When you're programming in a high level language you need a high powered editor. That's why we created a programmable full/split screen text processor. It lets you split the screen horizontally or vertically and edit two files at once. You can move text back and forth between two windows. You can also create your own macro commands from an assortment of over

100 predefined commands. The editor comes configured so that it works just like Wordstar but you can change it if you prefer a different keyboard layout. The editor is a great companion to our C compiler. Because they work so well together we want you to have both. To make sure you do, we're offering the editor for just \$15 when purchased with the C compiler.

ASM Utility \$10

The ASM utility disk allows you to link object files created by Microsoft's MASM or M80 assemblers. Lots of useful assembly language functions are included as examples.

ORDERS ONLY

1-800-523-9520

IN TEXAS

1-800-622-4070

Canadian Distributor

Saragay Software: 416-923-1500

NOT COPY PROTECTED

Editor \$ _____ (29.95)

C \$ _____ (39.95)

C & Editor \$ _____ (54.95)

ASM Utility \$ _____ (10.00)

TX Residents \$ _____ (6.125% sales tax)

Shipping \$ _____ (see below)

Total \$ _____

☐ Check ☐ Money Order

☐ MC/Visa# _____ Exp _____

Shipping Charges: (No charge for ASM Utility)

USA: \$5/Order

Canada: \$10/Order

Overseas: \$10/Editor • \$20/C • \$30/C & Editor

☐ PCDOS/MSDOS (2.0 or later)

☐ IBM PC Single Side

☐ IBM PC Double Side

☐ Tandy 2000

☐ 8 Inch

☐ Other _____

☐ CPM 80 (2.2 or later)

☐ 8 Inch

☐ Kaypro II

☐ Kaypro 4

☐ Apple (Z80)

☐ Osborne I SD

☐ Osborne I DD

☐ Morrow MD II

☐ Other _____

Name _____

Street _____

City _____

State _____

Zip _____

Country _____

Phone _____

MIX
software

2116 E. Arapaho
Suite 363
Richardson, TX 75081

(214) 783-6001

Ask about our volume discounts.

Bringing Up the 68000— A First Step

by Alan D. Wilcox

Adapted by permission of the publisher, Prentice-Hall Inc., from the forthcoming book Designing and Troubleshooting a 68000 Microcomputer System by Alan D. Wilcox, available 1986. No part of this adaptation may be reproduced, in any form or by any means, without written permission from the publisher.

There are two ways to bring up a new 68000 microcomputer system: the hard way and the easy way. The hard way is to use the traditional approach of designing the hardware and then using a development system along with test software and some in-circuit emulation. Given enough hours of testing and correcting problems, the 68000 system has a good chance of running successfully. In contrast, the easy way is to design, build, and test the hardware module by module using the 68000 as a free-running processor.

The impact of the free-running technique on hardware development is quite startling. The 68000 kernel shown in Figure 1 (page 61) can be made to run so easily that a logic probe can test it. You don't need to use sophisticated digital development tools such as a logic analyzer or a development system with an in-circuit emulator. If troubleshooting is necessary, you need only a common dual-trace oscilloscope.

Free running the 68000 means that the processor is allowed to execute a do-nothing instruction continually. This is accomplished by breaking the

The impact of the free-running technique on hardware development is startling. The 68000 kernel can be made to run so easily that a logic probe can test it.

normally closed loop between the 68000 and its memory, as shown in Figure 2 (page 61). Instead of carrying program instructions from memory, a one-word instruction (call it a NIL instruction) can be jammed onto the data bus. (The mnemonic NIL is not part of the 68000 instruction set per se; I coined it as a simple expression of the instruction used for free running.) Thus, when the 68000 reads the data bus for an instruction, it fetches the NIL word, executes it, increments the address, and reads the next NIL. This cycling repeats over the entire 16-megabyte address range; when the processor reaches the end of the 16 megabytes, it simply starts over again.

The strategy for bringing up the 68000 is to design, build, and test the 68000 kernel shown in Figure 2. Next, design and build one additional module, connect it to the kernel, and test it while the processor is free running. Add yet another module and test it while free running. You can free run the 68000 all the way through the construction of a complete CPU board. In fact, if a processor

board fails, you can usually free run it to help speed troubleshooting. The only part of the system that you cannot test easily while it is free running is the data bus itself, because the NIL instruction is forced on the bus.

The Steps

Several steps are involved in bringing up the 68000 using the free-running technique. The intent of this article is to describe the necessary first step: how to get the kernel running. Once you have the kernel in operation, the rest is fairly straightforward. Here is a brief overview of the entire scenario to complete a working 68000 CPU board:

1. Bring up the kernel. Design, build, and test the power system, the 68000 clock and drivers, the reset and halt module, and the 68000 module.
2. Add a wait state and data transfer acknowledge (DTACK*) generator module.
3. Add RAM and EPROM decoding circuits, connect address and control bus circuits.
4. Write a simple looping program for a pair of EPROMs. Remove the NIL instruction and close the broken loop between the EPROMs and the 68000 data bus. The processor should now be able to read its stack and program counter vectors from the EPROMs and execute the loop program.
5. Add the RAM connections to the data bus. If the 68000 is still running successfully with the simple loop program, add more code to include reading and writing RAM. If the code is a very tight loop, an oscilloscope will synchronize easily and you can use it to check the timing of the various control lines to all the memory in the system so far.

Alan D. Wilcox, 60 South Eighth St., Lewisburg PA 17837

6. Modify the reset and EPROM-control circuits so that the EPROMs do not have to be decoded at address 0 except during reset. Normally, the low memory addresses should be RAM so that exception vectors can be altered dynamically; EPROMs should be elsewhere.

7. If at least 4K of RAM has been decoded starting at address 8, and the EPROMs are decoded for 0 to 7 and \$8000 to \$BFFF, then the system can use the Motorola TUTOR EPROM set. When you restart the system, assuming that it does not halt, you can use an oscilloscope to see the activity on the various processor lines while the monitor waits for a console keystroke.

8. Add a 6850 ACIA decoded at \$010040 to serve as a console port and test it with the TUTOR EPROM set. Run various memory-testing commands and scope loops to check operation of the new system.

The First Step

As stated earlier, the first step is to bring up the kernel in the free-running mode. It seems a bit overwhelming when you first try doing it, but it really is quite simple. Unless you have made a wiring error, the 68000 is virtually guaranteed to come alive and begin executing the NIL instruction. To bring up the kernel, you need the power system, the clock and drivers, the reset and halt module, and the 68000 module.

The size of the power supply depends on the nature of the system and what loading it will have in the final configuration. In my own case, I intended to use the 68000 processor board in an IEEE-696 (S-100) system, so I needed on-card regulation from an 8-volt system supply. A common 7805 circuit was adequate for the processor and its RAM and EPROMs; I used a second 7805 circuit for the rest of the LS-TTL logic on the CPU board.

Watch the Motorola data manual for footnotes. In the case of the 68000, although the data indicates a power requirement of 300 mA or so, that is not the whole story. The fine print at the bottom of one page casually mentions that the 68000 might require a peak current of some 1.5 A. Make sure the power supply can handle the peak current without falling out of regulation. Likewise, power and

Address	Data	Program
00 0000	0000 0000	ORI.B #0,D0
00 0004	0000 0000	ORI.B #0,D0
00 0008	0000 0000	ORI.B #0,D0
00 000C	0000 0000	ORI.B #0,D0
00 0010	0000 0000	ORI.B #0,D0
.	.	.
FF FFFC	0000 0000	ORI.B #0,D0.

Table 1

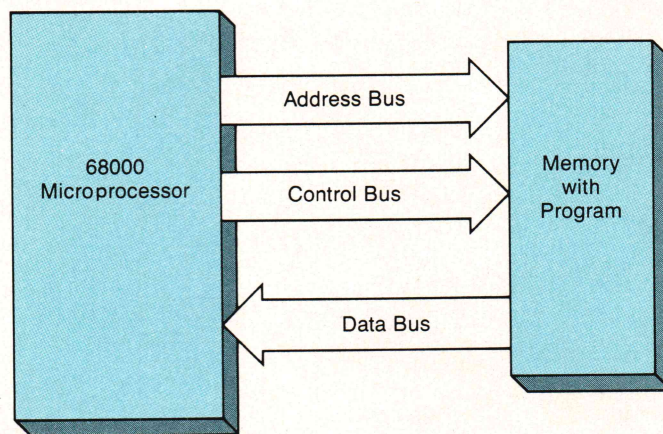


Figure 1: The 68000 kernel is the essential hardware for program execution.

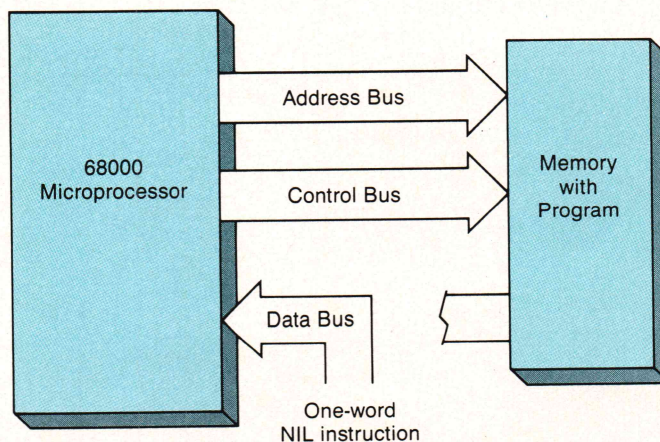


Figure 2: To free run the 68000, the normal feedback path from memory is disconnected, and a NIL (do-nothing) instruction is substituted.

ground leads to the 68000 need to be heavy, say #24 wire rather than #30 wire-wrap wires. Locate bypass capacitors close to each of the power and ground connections.

You can design and build the clock

circuit using a crystal, some resistors and capacitors, and a 7404 or similar; doing this is hardly worth the effort, though. For prototype work, being able to change the clock frequency easily without redesigning the circuit is a distinct advantage, so using a DIP oscillator is appropriate. I used a 6-MHz oscillator in my S-100 proto-

type to keep within the bus specification; only after I had finished the system did I run it up to 10 MHz and later to 12 MHz.

Also, you will use both the clock signal and its complement in the final circuit design. The complement clock could be derived from a 74LS04, but that would introduce a skew be-

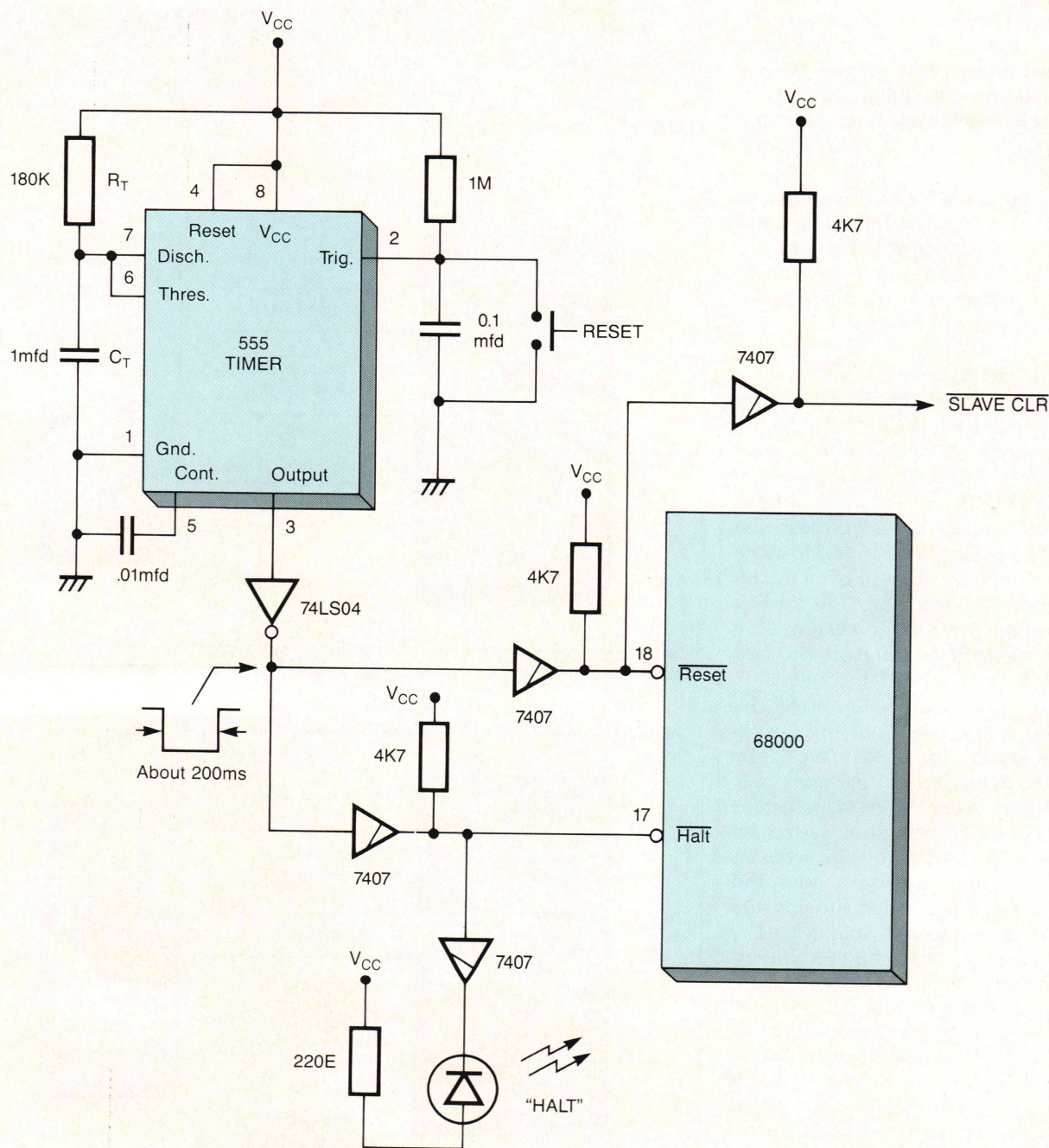


Figure 3: Circuit diagram of a simple power-up and reset timer circuit for a 68000 processor. Note the use of open-collector devices on the bidirectional HALT* and RESET* controls of the 68000.

C Programmers:

Consider 104 Ways To Be More Productive

If you find and choose the right development software, you can: cut development effort, make impractical projects feasible, and eliminate unproductive, frustrating aspects of programming.

Confused? We'll help you sort thru the huge number of alternatives. Call for comparisons or information.

Learn C Programming Only \$125

"Introducing C" Interpreter

Computer Innovations has done it again! This interactive implementation is combined with a full screen editor and a thorough, self-paced manual.

You can develop programs faster by getting immediate feedback. Programs will start instantly upon your command. There is no need to wait "for compile and link."

Introducing C includes demo programs, powerful C language interpreter, complete C function library, full screen editor, color graphics, and C language compatibility. PCDOS \$125

Shorten Development Time, Cut Frustrations

BRIEF, The Programmer's Editor

Compile within BRIEF; use autoindent; "templates", C specific error support... use windows, UNDO, and multiple large files.

But edit YOUR WAY.

Every feature you'd expect and more are integrated elegantly — and it can be modified by you.

You deserve:

"...the best text editor you can buy." — John Dvorak, InforWorld, 7/8/85

"...the best code editor..." David Irwin, Data Based Advisor, 8/85

PCDOS \$Call

Fast File Access with Source C-Index +

C-Index + contains a high performance ISAM, balanced B + Tree indexing system with *source* and *variable length* fields. The result is a complete data storage system to eliminate tedious programming and add efficient performance to your programs.

Features include random and sequential data access, virtual memory buffering, and multiple key indexes.

With *no royalties* for programs you distribute, full source code, and variable length fields C-Index + fits what you are likely to need.

Save time and enhance your programs with C-Index +. MSDOS \$375

Add Communications Features to Your Programs

Greenleaf Comm Library

Greenleaf now enables you to communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt driven system. This eliminates the extra function of separately calling up the communications program.

Included are 3 library/object files, 220 functions; 230 page manual, complete source code, library tailor-made to suit compiler and memory, Hayes-compatible modem commands, and a complete sample file transfer program. MSDOS \$169

First Aid for C Programs C ToolSet

Save time and frustration when analyzing and manipulating C programs.

DIFF and CMP - for "intelligent" file comparisons.

XREF - cross references variables by function and line.

C Flow Chart - shows what functions call each other.

C Beautifier - make source more readable.

GREP - search for patterns.

There are several other programs for converting and printing programs.

Portable. Full source code.

CPM, MSDOS \$135

SORT/MERGE Files for Clean, Fast Maintenance with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added. OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, FTN, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, Btrieve files
- 1 to 10 files Input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS \$90.

Get File Access with TIGHTER Control than an ISAM or a Sort. db_VISTA Data Management

6 files updating at once, entering data with validation...with no delays! Multiuser access at the same time. db_VISTA can do it.

db_VISTA makes practical sophisticated applications that cannot be handled with reasonable performance by a "relational" or "flat file" system. But, if you want a simple ISAM, it is included too.

Full source, no royalties and "normal" indexed file management are part of db_VISTA. Get more for the price of only an ISAM.

You can minimize data stored and access records even faster and more logically than just using indexes. Example: address and transaction data should not require redundant storage of customer names or numbers. Use pointers. Related data fields point to other related groups — the "network model" of data.

Use db_VISTA as a "normal ISAM" or save programming time, access time and file size. Lattice, C86, Williams, Desmet, Microsoft C.

MSDOS Multiuser source \$945, Object \$445

Single user source \$445, Object \$169

Unix, Xenix, and MacIntosh also available. Call for details.

Call for details, comparisons, or for our "C Extras Packet" with over 50 pages of information about C support products

THE PROGRAMMER'S SHOP

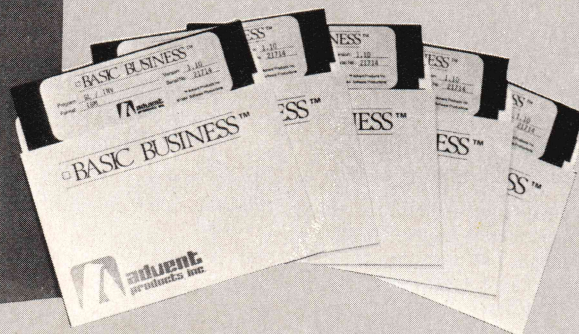
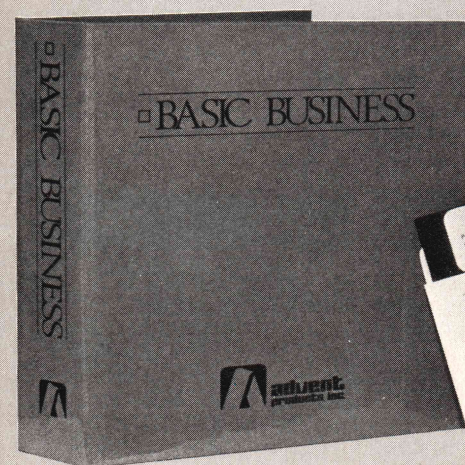
The programmer's complete source for software, services and answers

128-DC Rockland Street, Hanover, MA 02339 (617) 826-7531 (800) 421-8006

Ask about COD and PO's. All formats available. Prices subject to change. Names of products and companies are generally their trademarks.

Circle no. 141 on reader service card.

TIMELY REPORTS, ACCURATE RECORDS, RELIABLE INFORMATION.....



Complete accounting
software package, only

\$ 89⁹⁵

.....this is ☐ BASIC BUSINESS

Price is nice - and so is a long list of features - but when you choose a software package to automate your accounting, don't lose sight of the basics of good business record keeping. Basic Business, an all-in-one accounting software package consisting of General Ledger, Accounts Receivable, Accounts Payable, Payroll, Inventory Control, Sales Order Processing, and Purchase Order Processing. It offers an excellent price and extensive feature list without sacrificing these basic business principles:

Full Audit Trails - to give you complete confidence in the accuracy of your data and provide crucial backup information when you need it. And double entry accounting keeps your books in balance.

Complete Integration - provides efficient processing for all your business transactions, updating all of your accounting records instantaneously. You only enter the transaction once, and all supporting modules are updated automatically.

Flexibility - Basic Business can be adapted to your way of doing business, including balance forward or open item accounts receivable, departmental or consolidated general ledger, variable aging periods and easy entry of manually written checks and voids.

Extensive Reporting - each accounting module provides complete reports, including master lists, transactions, journals, statements and forms. In addition, all data files are compatible with Ashton-Tate's dBase II and dBase III, for the ultimate in custom reporting capability.

With some accounting software, even packages costing hundreds of dollars more, you must make detailed estimates and complex calculations for the maximum number of customers, transactions, inventory items, etc., before using the system. Then, when your business grows to exceed these original estimates, you must repeat this process.

Not with Basic Business! All data files are automatically initialized when you install the system on your computer. Files can grow dynamically as your business increases and are usually limited only by the amount of you exceed your original estimates.

FORMS ARE NO PROBLEM! Basic Business uses standard forms for invoices, statements, checks, purchase orders, etc., which may be ordered with your company name, address and logo imprinted.

Basic Business is one of a family of accounting and business software packages, and has sold previously for several hundred dollars per module. It has been improved, updated and re-packaged to sell at a market-busting \$89.95 for all seven modules. A Point-of-Purchase module, which controls an electronic cash drawer and allows direct entry of transactions from your sales counter is available. Also a dBase file format program is available for importing Basic Business data files into dBase II/III for custom report generation or other special uses.

SOURCE CODE AVAILABLE

Do you have a distinct accounting problem that off-the-shelf software won't handle? Special forms or statements? Don't write your own accounting system from the ground up - start with Basic Business. Call (714) 630-0446 for all the details on source code licensing.

Basic Business can go to work for you today and is available for most popular MS-DOS (IBM and compatibles) and CP/M-80 personal computers. Compare our price, features and attention to detail. There is only one choice... it's Basic Business.

Basic Business	\$89.95
Point-of-Purchase module	\$99.95
dBase II/III file formats	\$19.95

Minimum hardware Required for MS-DOS: 128K memory, two 360K floppy drives (hard disk recommended for Sales and Purchase Order Processing), 132 column printer, MS-DOS (or PC-DOS) version 2.0 or later.

Minimum Hardware Required for CP/M-80 computers: 80 x 24 character display terminal, 64K memory, two 360K disk drives (hard disk recommended for Sales and Purchase Order Processing), 132 column printer.

Circle **no. 176** on reader service card.

SERVICES

File Transfer Service: Advent provides a service beyond the ability of any format conversion software! We can transfer files between MS-DOS/PC-DOS, CP/M and other operating systems in 300 different 3 1/2", 5 1/4" and 8" formats. Includes Apple and Mac, Apricot, Data General One, Kaypro 2000, Eagle, Epson QX-10 & PX-8 (ROM), HP-150, and North Star computers.

ENGINEERING SOFTWARE

ACNAP: A stand-alone Electronics Circuit Analysis Program for use with passive and active circuits consisting of resistors, capacitors, inductors, transistors, op-amps, FETs, etc. Features menu driven and very fast processing times with circuits saved to disk for later use or editing.

ACNAP (CP/M & MS-DOS) \$69.95

DCNAP: Stand-alone DC circuit analysis program for use with passive and active circuits containing resistors, voltage sources, independent and dependent current sources. Fast, menu-driven program with circuit saved to disk for later use or editing.

DCNAP (CP/M & MS-DOS) \$69.95

Plotpro: Scientific graph printing program. Prints on 80 or 132 column printer. Create linear, semi-logarithmic, and full logarithmic plots with one or two Y axes in auto or forced scale.

Plotpro (CP/M & MS-DOS) \$69.95

SPP: This Signal Processing Program contains an integrated set of routines which analyze linear and non-linear systems and circuits and their effects on user specified time domain waveforms. Based on a 512 point Fast Fourier Transform and its inverse. Linear processing is in frequency domain and non-linear processing is in time domain.

SPP (CP/M & MS-DOS) \$69.95

SOFTWARE UTILITIES

Autodiff: File difference detector. This program finds insertions, deletions, and changes between any two files. Autodiff can mark the file, display, or print the differences, and more!

Autodiff (CP/M) \$29.95

CP/M DateStamper: Automatically stamp your files with the date it is created, last read, or modified. Works without a Real Time Clock, or with many clocks currently on the market. Utilities are included to allow copying, erasing, or renaming files based on time and date. A time logging utility is included to record computer usage for business/tax purposes.

DateStamper (CP/M) \$49.95

Media Master ++: Read and Write up to 75 CP/M, MS-DOS & TRSDOS disk formats on your IBM or look-alike computer. ZP/EM program is bundled with Media Master to allow CP/M programs to run directly on your MS-DOS computer. An \$80.00 value.

Media Master + \$59.95

Pack and Crypt: Two program set. Pack compresses and expands files on disk to save space. Crypt encodes files to provide security for sensitive data. Both are ideal for use with modem transfers.

Pack and Crypt (CP/M & MS-DOS) \$24.95

Sidekick: One of the most popular programs ever written. Use Sidekick as a calculator, notepad, appointment calendar, auto dialer, ASCII conversion table and much more. On-line help if you forget any of Sidekick's many functions.

Sidekick (MS-DOS) \$54.95

SmartKey II: New Release! Same great time saver as the original, and allows compiling of definitions you set up with your word-processor! Makes every software program you use easier. Can reduce keystrokes by more than 50% by redefining any key on your keyboard to be any combination of characters or commands that you desire.

SmartKey II \$49.95

SmartPrint: A powerful add-on to SmartKey, SmartPrint is a versatile writing tool designed to give you full access to your printer's features such as wide, bold, condensed, underlined, subscript, superscript, and more. Works great with programs like WordStar and others.

SmartPrint \$29.95

Uniform: Your Computer can read and write up to 80 CP/M, MS-DOS / PC-DOS & TRSDOS disk formats. Versions available for most popular CP/M and MS-DOS computers. Specify your host computer when ordering.

Uniform (CP/M & MS-DOS) \$69.95

XTREE: Directory maintenance program that graphically displays subdirectories and filename paths. Complete control of your directory including delete, rename, view, list or show. A must for your IBM or compatible.

XTREE (MS-DOS) \$49.95

Super Zap: Disk patch and dump program. If you have used DU, you will love this menu driven marvel!

Super Zap (CP/M) \$24.95

ZP/EM: Run almost any CP/M program on your IBM or clone. Use with Media Master or Uniform to allow programs on CP/M disk formats to run directly on your IBM or compatible computer.

ZP/EM (MS-DOS) \$39.95

FX, QX-10, PX-8 - Epson Corp; CP/M - DRI; MS-DOS - MicroSoft; PC-DOS - IBM Corp; dBASE II & dBase III - Ashton-Tate; WordStar - MicroPro; UNIX - Bell Laboratories; Apple - Apple Computer Inc.; Basic Business - Advent Products Inc.

PROGRAMMING LANGUAGES

C/80 Ver 3.1: Full featured C compiler and runtime library. One of the fastest on the market. Mathpak is included for true 32 bit floating point and signed integers.

C/80 Ver. 3.1 (CP/M) \$79.90

C/NIX: Operating System Enhancement for CP/M. C/NIX gives your system many features in the UNIX OS such a hierarchical directory, I/O redirection, "pipes" & "filters" and command files. Uses only 2.3K of TPA and 42K of disk. Requires CP/M 2.x.

C/NIX \$59.95

LISP/80: Experiment with the artificial intelligence language. Based on the INTERLISP dialect, LISP/80 offers over 75 built-in functions, including file I/O, and string operations. Complete with 36 page manual and demo programs.

LISP/80 \$39.95

Toolworks C: This compiler is a complete subset of C. The two-pass compiler produces relocatable object files (.obj) which are compatible with the MS-DOS LINK program. Mathpak is included for true 32 bit floating point and signed integers.

Toolworks C Compiler (MS-DOS) \$79.90

Turbo Pascal: Borland version 3.0. The best Pascal compiler on the market.

Turbo Pascal (CP/M & MS-DOS) \$69.95

Turbo Toolbox: Set of 3 utilities for use with Turbo Pascal.

Turbo Toolbox (CP/M & MS-DOS) \$54.95

Turbo Tutor: Teaches step-by-step how to use Turbo Pascal.

Turbo Tutor (CP/M & MS-DOS) \$34.95

Turbo Graphics: Provides full graphics management for producing windows, pie and pie charts, circles and other geometric shapes with Turbo Pascal.

Turbo Graphics (MS-DOS) \$54.95

TEXT EDITING

Punctuation & Style: Improves your writing by catching unbalanced quotes, parentheses and brackets, improper abbreviations, capitalization, sentence structure, much more. It's like having your own copy editor!

Punctuation & Style (CP/M & MS-DOS) \$125.00

Word Finder: This powerful 90,000 word Thesaurus allows you to select the best word for the application. Works inside WordStar for greater ease of use. Instantly searches its dictionary, then displays synonyms, and automatically deletes the "wrong" word and replaces it with the "right" word. Requires 380K disk storage.

Word Finder (CP/M) \$79.95

Wordpatch: Print files with tiny, compressed, wide, or wide compressed type faces. 5 sizes of italic, real superscripts and subscripts, and 6, 7, and 8 lines per inch spacing. No new print controls to learn. Supports most popular dot matrix printers. A must for WordStar users!

Wordpatch (CP/M & MS-DOS) \$49.95

The Word Plus: The ultimate spelling checker. Not only finds misspelled words but shows you correct spelling options, shows the word in context, allows you to build dictionaries of special words you use, and much more.

The Word Plus (MS-DOS) \$150.00

HARDWARE & SUPPLIES

Finger Print "Letter Writer": Unleash your Epson FX series printer. Add near-letter-quality print, IBM and/or Apple Graphics printer emulation, plus 16 other print functions! Three replacement chips quickly fit inside Epson FX series printers. Easy installation. Does not void printer warranty.

Finger Print "LetterWriter" \$79.95

Diskettes, Double Density:

Maxell 10-pack w/ storage box: 3M box of 10:

Single Sided \$19.95 Single Sided \$22.95

Double Sided \$23.95 Double Sided \$26.95

Economy Diskettes: package of 25 including tyvek sleeves.

Single Sided \$29.50

Double Sided \$31.25

Call or write for our FREE catalog

All items are warranted for 90 days. 30 day money back guarantee if not completely satisfied. Guarantee for software applies only if diskette seal is intact. Visa and MasterCard are welcome. Please add 2.00 freight per total order and 2.00 for COD orders. California residents please add 6% sales tax. Prices, availability and specifications subject to change without notice.

CALL TODAY National
California

(800) 821-8778
(800) 521-7182

Hours: Mon - Fri 8 am - 5 pm PDT

DEALER INQUIRIES WELCOME.



advent
products inc.

3154-F E. La Palma Ave
Anaheim, CA 92806
(714) 630-0446

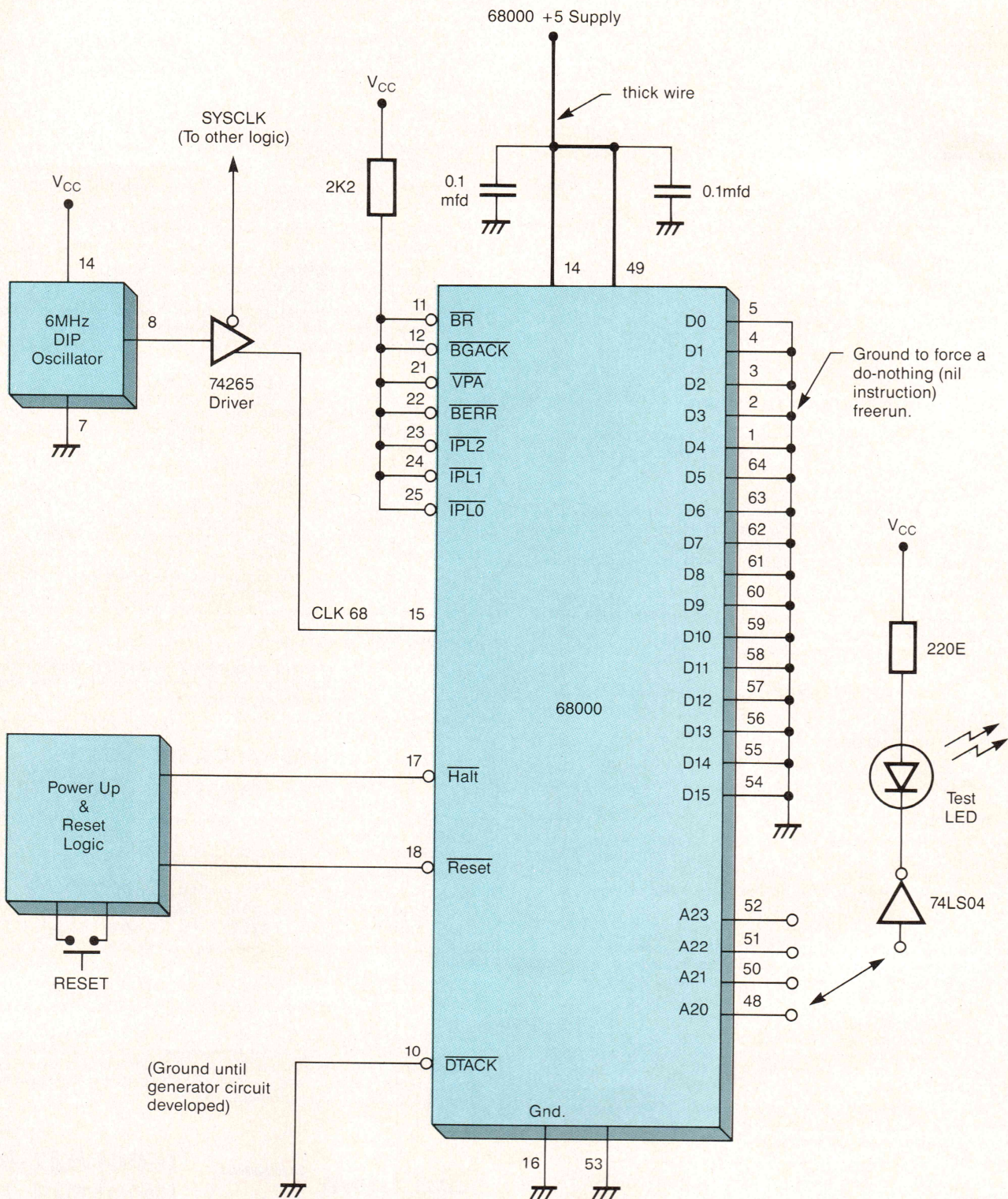


Figure 4: Circuit diagram of the minimum 68000 system for free-run test

Complete your Heath/Zenith system with a subscription to *Sextant*!

SEXTANT

The Independent Magazine for Users of Heath/Zenith Microcomputers

Whether you use an **H/Z100**, **H/Z89**, **Z90**, **H88**, or **H8**—you'll find articles in every issue which apply to your system.

Explore **CP/M**, **HDOS**, **Z-DOS**, and **MS-DOS** capabilities in *Sextant*. Applications, programs, compatible hardware and software, and the latest developments in the Heath/Zenith community are among the topics covered in every issue of *Sextant*.

Regular *Sextant* features include:

- "How-to" articles to help you enhance your system. Read how other users have altered their

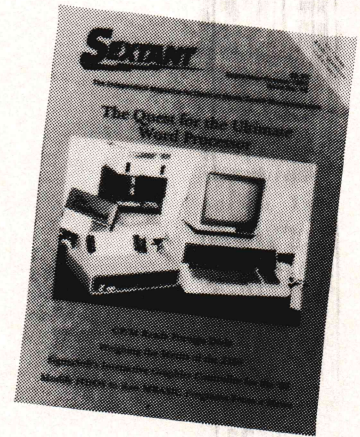
systems to suit their needs—and how you can too.

- **Reviews of products** from Zenith and independent suppliers, which can help answer your questions about hardware and software purchases you're considering.
- **Short program listings—including utility programs and games—**which you can use immediately.
- **Advertising by independent suppliers** who support Heath/Zenith systems. In *Sextant* you'll find many products advertised

that you just won't find anywhere else.

- **Coverage of community affairs.** You'll read about **major events and personalities** in the Heath/Zenith user community, and what effect they could have on you.

You can't afford to miss all the information packed into each issue of *Sextant*. **Your Heath/Zenith system just isn't complete without a *Sextant* subscription!**



Get the information you need to get the most from your system!

Start your subscription today!

Call Toll Free: 800/341-1522



(DATATEL 800™, for orders only, Monday-Friday 8 a.m.–9 p.m., and Saturday 9 a.m.–5 p.m., Eastern Time)

Or mail this coupon to:
Sextant, Dept. DD1
716 E Street, S.E.
Washington, DC 20003

Please allow 6–8 weeks for delivery of your first issue.

Your satisfaction with *Sextant* is completely guaranteed. If at any time you're not completely satisfied, just let us know and your money will be refunded—even for the issues you've already received.

Send me:

- ☐ 12 issues for \$29.91 (\$34.50 to Canada)
- ☐ 6 issues (1 year) for \$14.97 (\$17.25 to Canada; \$21.00 overseas via surface mail, \$35.00 via air mail)
- ☐ Payment enclosed (Checks must be in U.S. dollars payable on a U.S. bank.)
- ☐ Bill me
- ☐ Charge my: ☐  ☐ 

Card # _____ Expires _____

Name _____

Address _____

Sextant, Dept. DD1, 716 E Street S.E., Washington, DC 20003

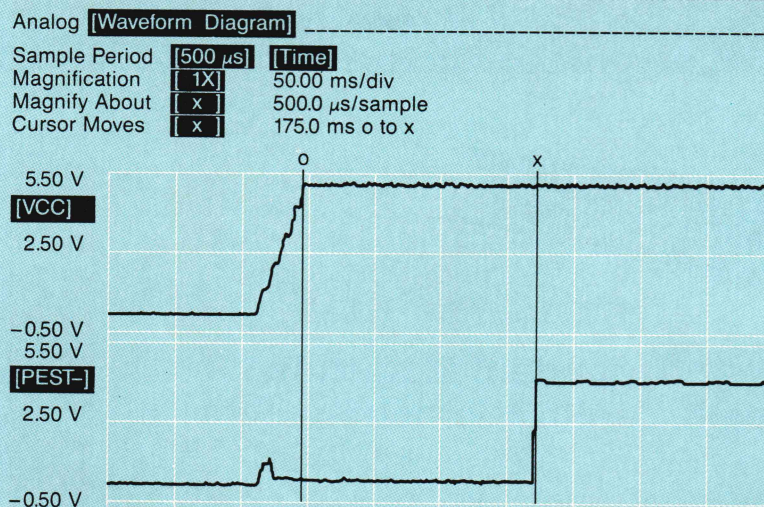


Figure 5: Power-up performance of the 555 timer circuit. On power-up, the 555 timer with the parts given in the schematic provides about 175 ms RESET* to the 68000.

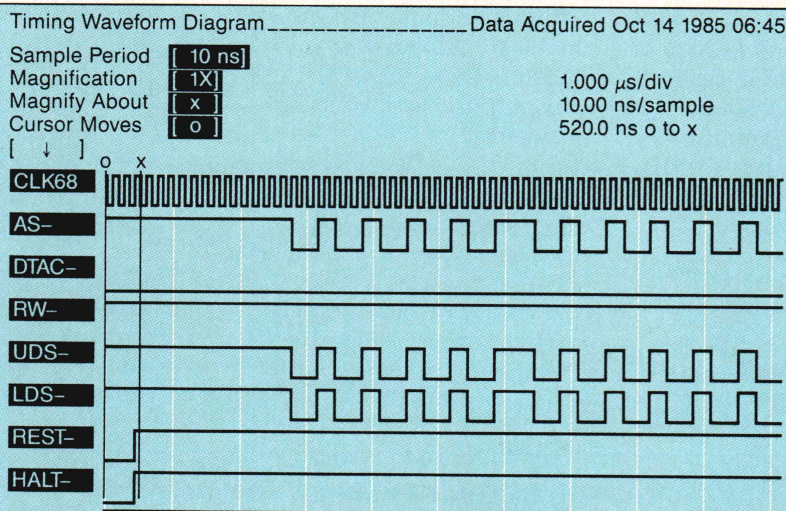


Figure 6: Typical free run starting from a complete RESET* and HALT* asserted low. The clock is running at 6 MHz. DTACK* is grounded in this example.

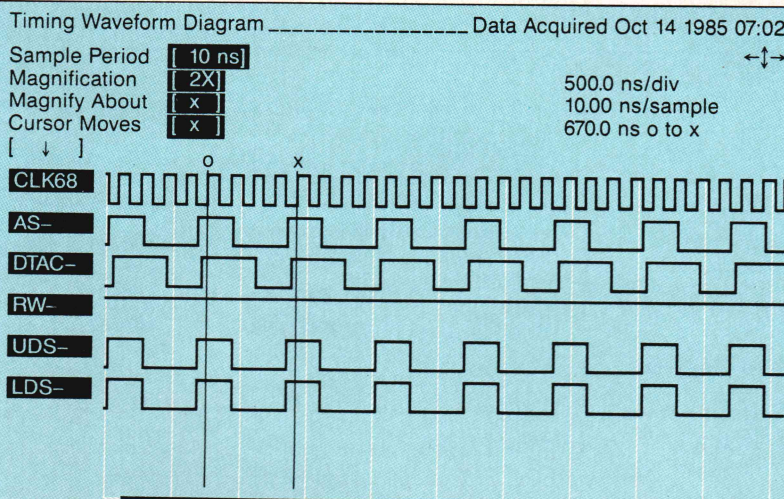
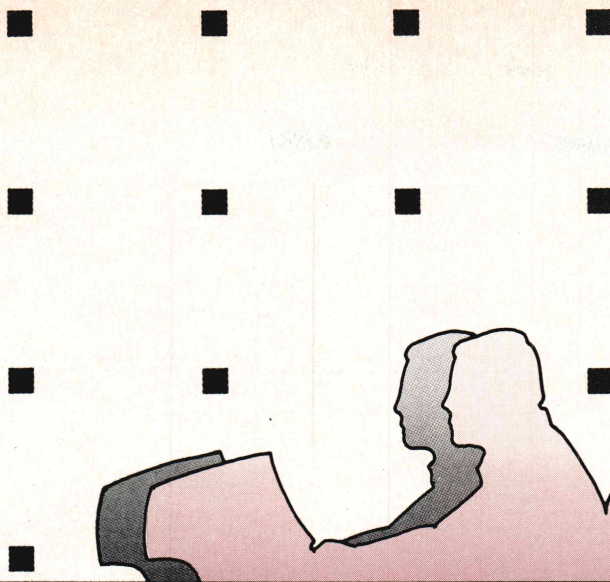


Figure 7: Typical free run with the DTACK* circuit enabled. The clock is 6 MHz, and there are no wait states inserted.



THE International UNIX* Event of the Year!

February 4-7, 1986 / Anaheim Convention Center / Anaheim, California

UniForum 1986, the International Conference of UNIX Users, has established itself as THE premier UNIX conference/trade show.

■ Over 200 major vendors exhibiting their newest UNIX-based hardware, software, systems, services and peripherals.

■ A complete tutorial and conference program. Multiple sessions target the connectivity of UNIX between the user and technical environments; the hardware and software technology for office systems and workstations; the interface between man and machine, machine and machine, and much more. Day-long tutorials provide intensive, focused material on specific subjects in UNIX...while the conference sessions highlight the latest developments in the continuing evolution of UNIX.

*UNIX is a trademark of AT&T Bell Laboratories.

■ **FREE** UNIX Introductory Workshops.

■ Birds-of-a-Feather impromptu sessions.

Call us NOW for your **FREE** Show-Only Badge *and* complete registration information on the dynamic conference/tutorial program.

The power and potential of UNIX are important to you. UniForum 1986 is THE computer event you can't afford to miss!



UniForumTM
The International Conference of UNIX Users

For Complete Information, Call:

800-323-5155

(In Illinois, Call: 312-299-3131)

Or Write:

UniForum
2400 East Devon Avenue
Suite 205
Des Plaines, IL 60018

Sponsored by



usr/group

The International Network of UNIX Users

Circle no. 120 on reader service card.

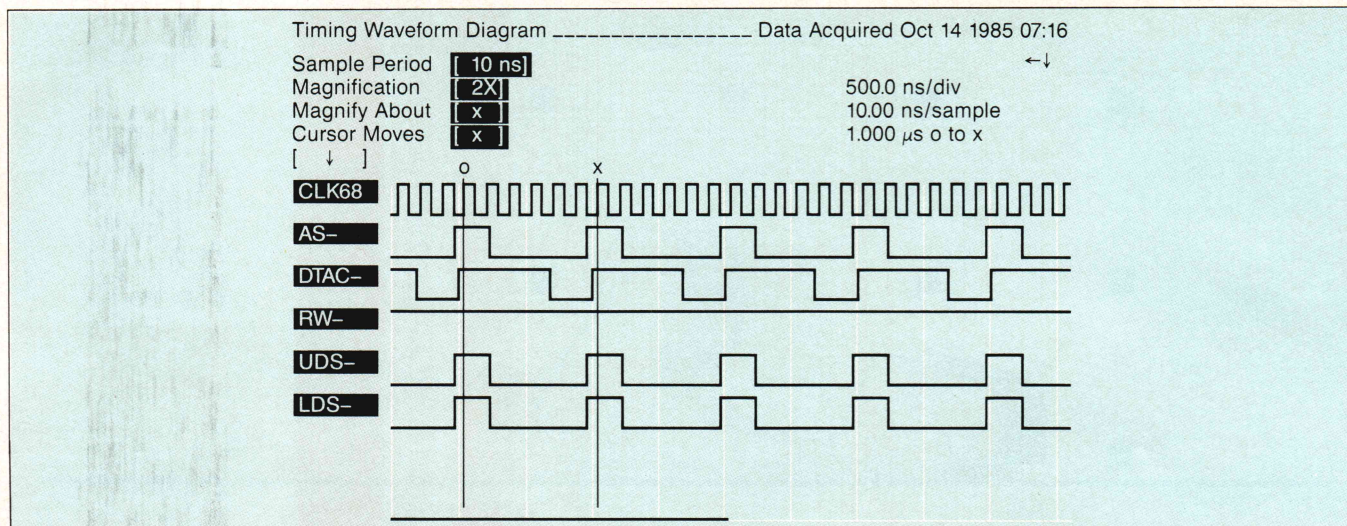


Figure 8: Typical free run with DTACK* enabled. This timing shows DTACK* delayed enough to cause two waits in each bus cycle.

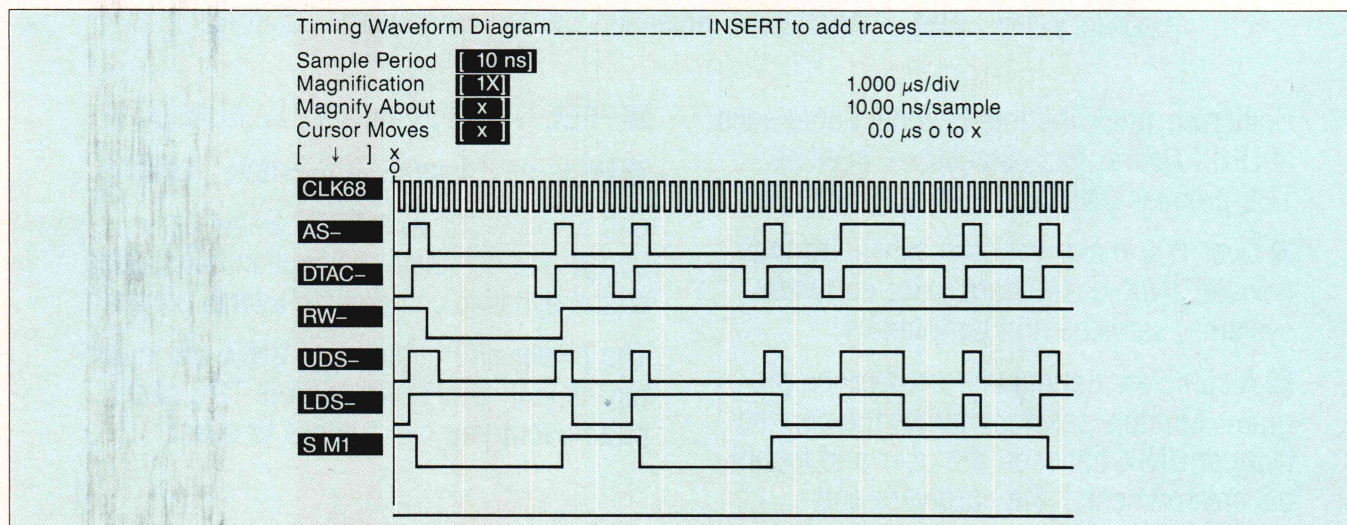


Figure 9: A view of the bus activity when the TUTOR EPROM set runs at 6 MHz. The CPU board was set for eight waits on I/O and three waits otherwise.

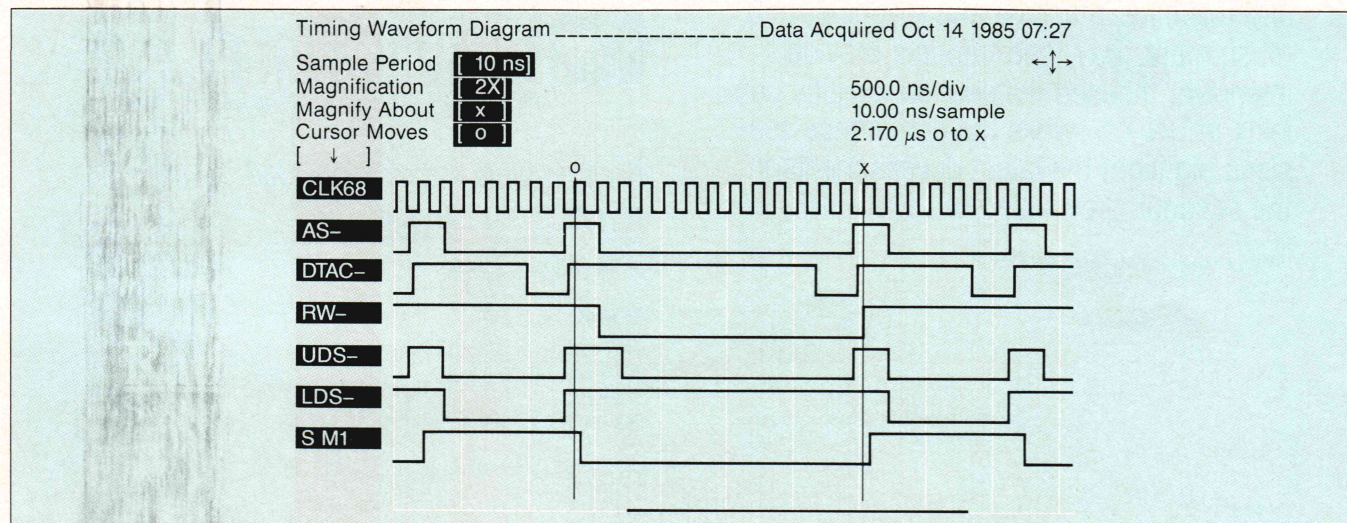


Figure 10: A closer look at the bus controls when TUTOR executes a write bus cycle

Now You Know Why **BRIEF** is **BEST**

"BRIEF, The Programmer's Editor, is simply the best text editor you can buy." John Dvorak, INFOWORLD 7/8/85

The Program Editor with the **BEST** Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)... is quite simply the best code editor I have seen."

Solution Systems™

"A bona fide Undo..."

Steve McMahon, BYTE 3/85

As Mark Edwards describes in DR. DOBB'S JOURNAL (11/85), "BRIEF has an outstanding undo facility. The default configuration allows the last 30 editing commands to be undone. This number can be raised to a maximum of 300 commands. Until you reach this maximum or run out of RAM, every command you issue can be undone. So if you make ten changes and then realize that the first one was an error, you can undo all the changes back to the mistake... Needless to emphasize, this facility can save endless grief."

No other editor has this capability.

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size – (even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

MONEY-BACK GUARANTEE

Try BRIEF (\$195) for 30 days – If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

Program Editing **YOUR** Way

A typical program editor requires you to adjust your style of programming to its particular requirements – NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key – even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion – **BRIEF IS BEST!**

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

Thinking of the C Language?

THINK COMPUTER INNOVATIONS

NEW!!

C86 VERSION 2.3 with Source Level Debugging Support

The C language has rapidly become the development language of choice for applications ranging from Operating Systems to Accounting Packages. WHY? Its structured approach and extreme portability make it perfectly suited to today's fast-paced environment.

Of all of the C Compilers available for PC/MSDOS, more programmers choose COMPUTER INNOVATIONS' C86. WHY? Because it's part of a COMPREHENSIVE family of C products with an unparalleled reputation for performance, reliability, and stability.

C86 2.3 C COMPILER

C for PC/MSDOS began with C86 and today it remains perhaps the most solid, stable C Compiler available. Even competitor's ads show C86 as a consistent top level performer in benchmark testing.

Version 2.3 offers a host of new features including source level debugging support and a 40% boost in compilation speed. Call for complete specifications.

COST: \$395 UPDATE TO 2.3: \$35 w/old diskettes NOT COPY PROTECTED CALL ABOUT VOLUME DISCOUNTS

LEARN C INTERACTIVELY WITH INTRODUCING C

Intimidated by rumors about the difficulty of learning C? Need to train your staff quickly? INTRODUCING C can help. INTRODUCING C combines a thorough, self-paced manual with a unique C interpreter to provide a fast, efficient method of learning C. Designed for both professional and casual programmers, it provides a comprehensive understanding of important C concepts such as standard K&R syntax and operators, full structures and unions, arrays, pointers, and data types. Requires IBM PC, XT, or AT with one disk drive and 192K bytes of memory.

COST: \$125 - NOT COPY PROTECTED

CI PROBE SOURCE DEBUGGER

Take advantage of C86 2.3 source level debugging support with CI PROBE. Cut down program development time and save money! CI PROBE is highly economical yet has the features of debuggers costing far more.

COST: \$225 - NOT COPY PROTECTED

C-TERP C86 COMPATIBLE INTERPRETER

The C-TERP INTERPRETER is a full K&R implementation that allows you to write code and execute it immediately without the compile and link steps. Once you have your program running with C-TERP you can compile the code (without alterations) with C86 for fast, efficient executable files. C-TERP requires 256K, 512K is recommended.

COST: C86 version - List Price: \$300, Special Computer Innovations Price \$250. Combined C86 & Lattice version - List Price: \$400, Special Computer Innovations Price \$350.

Start With Us, Stay With Us

Computer Innovations offers a complete range of products that let you enter the C environment and create applications with the most advanced set of development tools available. Unparalleled tech support assures that you're always at the height of productivity.

To order call: **800-921-0169**



COMPUTER INNOVATIONS, INC.

980 Shrewsbury Ave., Tinton Falls, NJ 07724 • (201) 542-5920

C-TERP is a trademark of Gimple Software. Prices and specifications subject to change without notice.

68000

(Continued from page 62)

tween the two clocks of some 10 to 15 ns, depending on loading. Although this amount of skew seems slight, it can cause severe timing difficulties when the clock speed gets above 10 MHz. The 74265 quad complementary-output element with a worst-case skew of 3 ns is a good selection; in my 12-MHz prototype, this selection has worked out well.

The reset and halt module has two basic functions. One task is to hold the 68000 HALT* and RESET* lines low for at least 100 ms on power-up. Its other function is to pull the same two lines low for at least ten clock cycles for a reset button press at any time after the power has been on.

The circuit in Figure 3 (page 62) provides a simple and reliable reset function for the 68000. It provides a reset pulse either on power-up or whenever you press the reset button. Open-collector devices are required because the 68000 HALT* and RESET* controls are both bidirectional. For example, the 68000 can itself drive the RESET* line to reset any peripherals if the software reset instruction is issued. Also, the 68000 can force the HALT* line low if the system cannot continue processing. A single "halt" LED connected as shown is valuable in helping bring up the processor for the first time.

The last module in the minimum system is the 68000 processor shown in Figure 4 (page 66). By now, you should have checked the power, clock, and reset modules for proper operation and connected them ready for the 68000. If the processor is wired as shown, it should begin free running immediately. On power-up the HALT light should flash briefly, and then the TEST light will begin flashing on and off.

Earlier I referred to my so-called NIL instruction. As you see in the circuit, the data bus is completely grounded so the NIL has an opcode of 0000. In the context of its use in free running, it acts like a no-operation or NOP. The 68000 does have a NOP opcode (\$4E71), but this NOP will not work as a free-running instruction.

A critical constraint on the opcode precludes using the NOP instruction in free running: Whatever is wired

to the data bus for the 68000 to read upon reset must be even. The reset sequence is this: The 68000 will do four 16-bit reads to get the initial SSP and PC vectors; then it will fetch its first opcode at the address in the PC. If the PC is not aligned on an even address, the 68000 detects an address error and immediately begins illegal-address exception processing. It tries to push its status on the stack at the beginning of the exception, but the stack is also an illegal address (the same noneven number as in the PC). The result is the fatal double bus fault that stops all processing and asserts the HALT* output.

The opcode 0000 does in fact correspond to a real instruction in the 68000 set. It is the mnemonic ORLB #0,D0, and it was selected for free running for two reasons: first, because it was even; and second, because connecting all grounds to the data bus was simpler than making sure one or two data lines had a logic 1 on them. When the instruction is considered in its free-running environment, the appearance of its memory is as shown in Table 1 (page 61).

You can calculate the execution time of this "program" easily. Each instruction takes eight clock cycles (two read bus cycles), so for a 6-MHz clock, the execution time is 8×167 ns or approximately 1.33 microseconds. A complete sweep through the entire 16 megabytes of the 68000 address range takes 1.33×4 megabytes or about 5.59 seconds. If you connect the TEST light to the top address bit, A23, it will be on for 2.8 seconds and then off for 2.8 seconds. I connected the TEST light to A20 permanently. It stays on for 0.35 seconds and off for 0.35 seconds—a reassuring flash rate during development work and not nearly as unsettling as a constant red HALT light.

Results

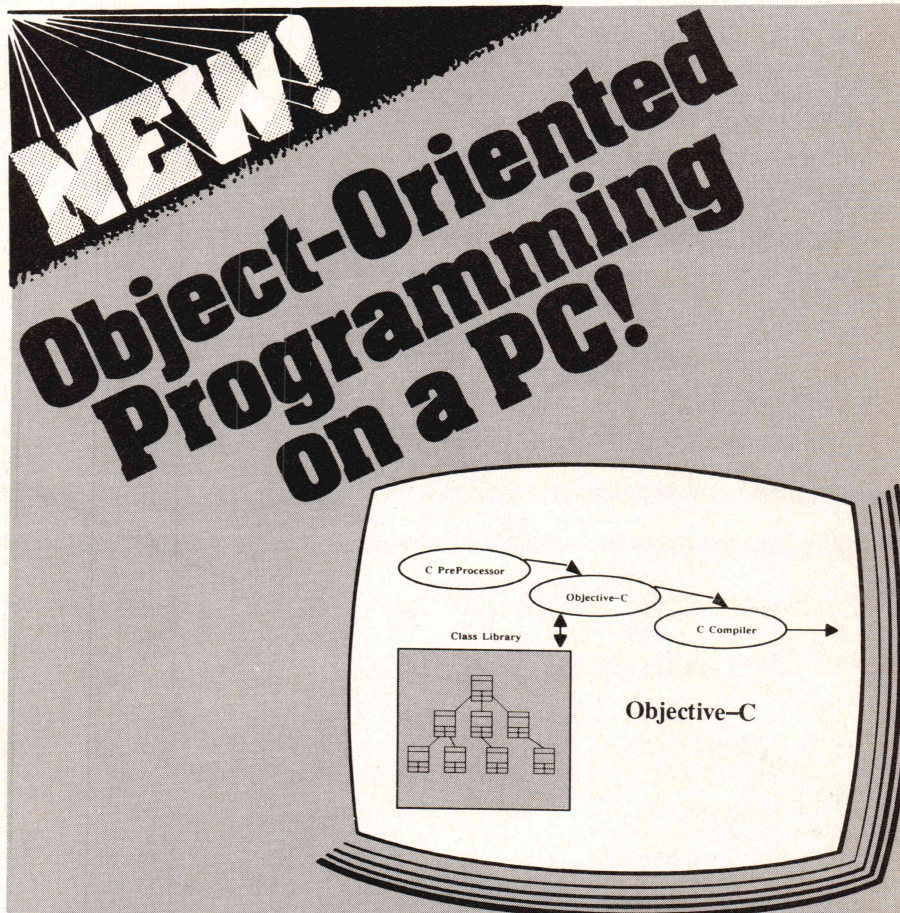
Figure 5 (page 68) shows the performance of the power-up timer circuit. The top plot is the main system power as it comes on and eventually regulates at 5 volts on the CPU board. About 175 ms after the supply voltage is valid, the RESET* and HALT* lines go high to successfully complete a full 68000 reset.

The effect of this reset operation is shown in Figure 6 (page 68). The last

two lines on the timing diagram are the RESET* and HALT* controls for the 68000. After its internal start-up time, the 68000 asserts its address strobe (AS*) and its data strobe lines (UDS* and LDS*) in the first read bus cycle. After four read bus cycles, the processor PC begins execution at address 0, as discussed above.

DTACK* is the asynchronous bus

control line that normally comes back from memory or peripherals to tell the 68000 to complete the current bus cycle. During the initial free run of the processor, there is nothing connected that will acknowledge a data transfer, so the control is grounded. The timing diagram shows this line at a logic low. The timing diagram also shows the read/



Objective-C™ is an object-oriented programming language and a fully documented library of reusable components...adding messages, objects and inheritance to C language. Applications written in Objective-C are fully compatible with other Objective-C compilers running under UNIX, VMS or AOS.

Objective-C provides the productivity of object-oriented programming, while

retaining the portability and efficiency of C. PPI also provides comprehensive technology transfer to insure that you, the programmer, fully understand this exciting new technology.

PPI's Objective-C compiler generates code, which requires the Microsoft Visual C compiler running under MS/DOS.

At \$500 Objective-C is affordable. Order today!

PRODUCTIVITY PRODUCTS INTERNATIONAL
27 Glen Road, Sandy Hook, CT 06482. (203) 426-1875.



Circle no. 140 on reader service card.

Learn and Use AI Technology In Your First Evening With TransPROLOG-PC

A complete *Prolog Interpreter, Tutorial, and set of Sample Programs:*

☐ **Modify and write Expert Systems.**

Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the TransPROLOG-PC authors and published in the March, 1985 issue of Dr. Dobb's Journal.

☐ **Understand Natural Language**

Use the sample program that produces a dBase DISPLAY command as output.

Programming experience is not required, but a logical mind is. Serious development of experimental systems is practical with TransPROLOG-PC. 1 or 2 pages in Prolog is often equivalent to 10 or 15 in C.

RECENT IMPROVEMENTS: MSDOS commands, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

☐ **Write Symbolic Math or Abstract Problem Solving Applications**

This is a complete Prolog program to convert from Fahrenheit to Centigrade: f_to_c(C,F):- C is(F-32) *5/9. Planning programs and games are included to help you learn.

☐ **BECOME FAMILIAR WITH PROLOG IN ONE EVENING.**

**ONLY
\$125**

Full refund if not
satisfied during
first 30 days.

**Solution
Systems™**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 152 on reader service card.

LEARN LISP

**Interactively and Write "Realistic" Programs
with TransLISP-PC for Only \$75**

*A "COMMON LISP" compatible Tutorial, Interpreter, Debugging, and
Pretty Printer plus a Fast, Full Screen Editor, Samples and Help*

☐ **Start Easily and Quickly:**

A complete, modular tutorial helps you learn LISP at your own pace. An integrated, interactive environment provides all of the elements needed to enter, modify, analyze and debug programs.

☐ **Natural Language, Expert Systems and Mailing List:**

Natural Language concepts are illustrated by a phone number retrieval program. Choose the best word processing program for you with the Expert System. File handling and typical data processing work are demonstrated by a Mailing List program.

☐ **Write Realistic Programs:**

Short examples and substantial programs of about 10 pages in length help you learn by modifying, studying and using the key concepts needed to write programs of 1000 lines or more.

☐ **The "COMMON LISP" Standard:**

TransLISP-PC includes a 230+ function subset of the "COMMON LISP" Standard. Use extras like the MSDOS interface and graphics. Or use "strict compatibility" to make programs written in TransLISP-PC, with no changes, work with other COMMON LISP systems like VAX LISP, GC/LISP or LISP Machine LISP.

Use and Modify the "Which Word Processor?" program to learn about EXPERT SYSTEMS.

Runs on any MSDOS or PCDOS Systems: Not copy-protected, TransLISP-PC is available in just about any 3", 5" or 8" format. PC compatibles can run TransLISP-PC with no installation procedure. 192K memory and 1 floppy drive are the minimums required.

ONLY \$75 For Beginners and Experienced Programmers

Full refund if not
satisfied during
first 30 days.

**Solution
Systems™**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 153 on reader service card.

68000

(Continued from page 73)

write control, R/W*, as constantly high because the 68000 only does a read bus cycle when it is free running.

Figure 7 (page 68) shows the free-running processor with a DTACK* generator in operation. Notice the o and x markers bracketing a single bus cycle. The normal read bus cycle has a total of four clock cycles. If DTACK* is delayed for two cycles, as shown in Figure 8 (page 70), then the bus cycle is lengthened and two "waits" are inserted into each bus cycle. When you interface memory or peripherals to the 68000, you can design each external module to hold back DTACK* until its unique timing requirements are met.

As an example, the timing diagram in Figure 9 (page 70) shows the system while *not* free running: It is executing the monitor program (TUTOR) and waiting for a keystroke. The system was set to provide eight waits for I/O read operations, nine for writes, and three waits otherwise. Figure 10 (page 70) shows a close-up look at the bus cycles. The lower timing line, marked sM1, is the S-100 bus status indicating an opcode fetch.

Summary

Bringing up the 68000 using the free-running technique is very different from the more traditional approaches to getting a processor running. You can see, though, just by the brief description of this first step in bringing up the 68000 kernel, that you do not need sophisticated equipment to get started.

There is more to be said about all the steps beyond this first free-running processor; no doubt many questions remain unanswered. From my experience, though, the understanding you can get from doing a free-running 68000 is very valuable, and it can help you go on to design and build a complete system successfully.

Availability

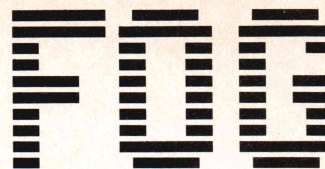
The TUTOR firmware is available directly from the author.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 4.**

Do you own a computer? Join
 Do you want to know more about using it? Join
 Do you find the manuals incomprehensible? Join
 Do you want access to lots of software? Join
 Do you need a technical support hotline? Join



FOG is a users group with 15,000 members from around the world who attend local group meetings at over 300 locations. Computer systems owned or used by members include all of the Osborne, Morrow, Kaypro, and IBM systems; the Zorba; the PMC MicroMate; the CompuPro System 10; the AMQTE; Epson QX-10 and PX-8; the Lobo MAX-80; and many more.

FOG was started in October of 1981 by a small band of early buyers of the Osborne 1 whose primary purpose was to organize a library of public domain software. A newsletter was quickly started to act as a focal point for the group's activities. As the market expanded and new manufacturers created competing models, these owners joined the group. The large number of excellent contributions to both the library and the **FOGHORN** has produced an extensive library of disks and a typeset (70 or more pages) monthly publication aimed at CP/M users. Recent interest in 16- and 32-bit operating systems resulted in the creation of a separate publication for those users. (All back issues for both publications are available for a nominal fee which includes shipping in the U.S.)

A network of BBS-RCP/M systems was started to increase access to the disk library. There are now about forty systems around the world in the network with more going up every month. Most of the systems are on line 24 hours per day. Phone numbers are listed in each issue of the **FOGHORN**. Registration is required under a system which allows **FOG** members to register on line.

FOG holds no meetings. Instead, meetings are organized by local groups. 140 of these groups have joined the **FOG** network, thus increasing the sharing of information, tips, problems and so on. Those local groups which opt to formally join the **FOG** network receive a portion of local member dues to assist with the cost of maintaining a local copy of the disk library.

The **FOG** library is organized on 170K disks so that members may order library copies in their own format. The library disks are carefully screened to be sure that each program works and is fully documented. Members who download from one of the RCP/M systems or copy disks at their local Affiliated Member Organization (supplying their own disks) do so at no charge. Those who find this inconvenient may order the disks from the **FOG** office. A printed library directory is available to help you find the programs you are seeking. A Starter Disk of essential utilities has been prepared for new members.

The **FOG** office in Daly City is open 10:00 am to 5:30 pm Pacific time. Technical support personnel are available to answer questions

or to guide you in finding a program or identifying the source of operating failure. Other staff members will refer you to your nearest local group or aid you in obtaining copies of the disk library and other materials.

Dues in FOG are \$24.00 per year. This entitles each member to a subscription for one of the publications each month as well as access to the disk library and RCP/M network. Members who choose to receive both publications may do so for \$42.00 (plus applicable postage charges) per year. Local group meetings are open to the public without charge although access to the disk library is restricted to the membership. The **FOG** Disk Library contains only public domain software. Piracy (the copying of proprietary software) is strongly condemned.

In the United States, the publications are normally mailed by non-profit bulk mail. (**FOG** is a corporation in the state of California and has obtained its non-profit, tax exempt status from both the state and federal governments.) For those members who live out of the country or who prefer first class delivery, additional postage must be added to the annual dues. See the chart below for details.

If you are interested in joining a self-help organization to increase your knowledge and the use of your computer, use the application below (or a copy of it). If you know of a local group which might be interested in joining the **FOG** network, please send all details (meeting dates and places, officers, and how interested local computer owners can join). We will send you an information packet on becoming an Affiliated Member Organization.

For your records, the address of **FOG** is P. O. Box 3474, Daly City, CA, 94015-0474. Please allow at least two months for the arrival of your first **FOGHORN** since bulk mail can take as much as nine weeks. (The post office says that it should only take about three weeks for non-profit bulk mail but some members on the East Coast have experienced longer delays.) A membership card will be processed within two weeks of the receipt of your dues.

ADDITIONAL POSTAL CHARGE CHART

The following is a list of additional charges **per publication**. If opting to receive both publications, please double the amount shown.

Canada & Mexico (First Class)	ADD...\$ 9.00
Members with U.S. addresses who prefer First Class delivery to Non-profit bulk mail ...	ADD...\$ 9.00
Central & South America, Caribbean, and Europe (Airmail First Class delivery)	ADD...\$12.00
Asia, Africa, & Far East (Airmail First Class)	ADD...\$15.00
Out of North America preferring surface mail — delivery not guaranteed	ADD...\$ 6.00

CUT HERE

Name: _____
 Company (if part of mailing address): _____
 Address: _____
 City: _____
 State: _____ ZIP or mail Code: _____ Country: _____
 Publication desired (circle one): **FOGHORN** (\$24/yr) **FOG** + (\$24/yr) **BOTH** (\$42/yr) **1st Class postage** (from chart): _____
 Home & work phones: _____
 Member of which local group? _____
 Computer type(s)? _____
 Modem type(s)? _____ Printer type(s)? _____
 Interests? Word processing Spreadsheets Data mgmt Education Business Modems Graphics Statistics Ham radio Engineering Genealogy Scientific
 Programming Language or Other interest: _____

Send this completed application **AND** your payment to: **FOG**

P. O. Box 3474
 Daly City, CA 94015-0474
 United States of America
 Phone: (415) 755-2000

Be sure to include any required postage surcharge.
Your membership card will be sent by 1st class mail.

COM: An 8080 Simulator for the MC68000

by Jim Cathey

Simulation time can vary widely, as some 8080 instructions aren't easily simulated with the 68000.

After I bought a 68000-based S-100 system, I found that I needed to run several CP/M 2.2 (alias CP/M-80) software packages, none of which were available in equivalent forms for my machine. I was faced with two options—either buying another processor board so that I could run these programs or writing an 8080 simulator. The software approach seemed infinitely preferable. This also seemed like an opportune time to find out just how fast my 68000 really was. So I set out to write an 8080 simulator (which I named COM because it interprets .com files).

In principle, writing a simulator is simple. You just set up a set of fake registers and start picking up opcodes and interpreting them. Unfortunately, it's the little details that get you. This simulator took about twice as long to write as I expected. As it is, it isn't perfect. In fact, it would slow down considerably if it were. However, most programs aren't bothered by the imperfections, and the speed difference would be significant because the simulation is already on the slow side of usable. I wrote COM to run as fast as possible.

The simulation speed is approximately that of a 1.4-MHz Z80 processor based on a sample assembly with MAC. (My 68000 system is an 8-MHz CompuPro/Morrow single-user hybrid running CP/M-68K 1.2, with 16-bit no-wait-state memory.) Simulation time can vary widely, as some 8080 instructions aren't easily simu-

lated with the 68000. MAC was chosen as a typical example program. LU, the public-domain library utility, is one of the worst performers. It spends vast amounts of time calculating CRCs. This instruction sequence isn't very efficient on the 8080 (using a C arithmetic library), and the simulation magnifies any inefficiencies. I didn't time LU for comparison, but I thought that the simulation had crashed because nothing seemed to be happening for long periods. Another poor performer is WordStar. I tried simulating it because it is so popular and because it gives the simulator a thorough workout. The simulation works, and it does so at an acceptable performance level most of the time.

About the Program

The program source is broken into four files. Listing One (page 104) is the first file, which contains the start-up, command-line, CP/M-2.2 simulation, and trace routines. Listing Two (page 112) contains opcode simulation subroutines and flag tables. The rest of the listings will be continued in the March issue.

The program starts out by prompt-

ing for the trace end points if that code has been included. (There are several conditional assembly trace features in COM.) It then builds the 8080 environment in a 64K buffer (biggest TPA yet!) and initializes the 68000 simulation registers, with the 8080's PC set to 100H into the buffer. It then calls a subroutine that loads the specified .com program into the buffer and transfers the 68000's second FCB to the 8080's first FCB and passes the remains of the command tail to the 8080's DMA buffer. If all goes well, the program then enters the main loop at label MLOOP. Here it fetches the next 8080 opcode and uses it to index into a table of 256 68000 subroutines (one per opcode—big, but fast) and jumps to the selected subroutine. Each opcode subroutine then picks up what parameters it needs and plays with the fake registers appropriately. Each subroutine then jumps back to MLOOP, which repeats the process. This continues until a service request is picked up or until an illegal instruction is found.

The service request used by the simulation is the HLT opcode (\$76). HLT is followed by a 1-byte parameter telling the 68000 which action to take. All BIOS/BDOS functions are implemented as service requests. After the service is performed, execution continues at MLOOP (or at the byte following the parameter—it depends on your point of view). Refer to the 8080 Environment section for more details on how service requests are used by the 8080's BDOS/BIOS.

Register dumps are caused by illegal opcodes or are done during a trace. They are easily interpreted (registers S0-S3 are the top four en-

Jim Cathey, ISC Systems Corp., TAF-C8, Spokane, WA 99220.

tries on the stack) as the current instruction is also disassembled. Illegal opcodes terminate the simulation after the register dump.

Flag simulation is done with two tables. Because any 8080 logical operation that sets the parity flag also clears the carry bit, these flag results are based solely on the value in the accumulator. The simulator uses a 256-byte flag lookup table for these operations. Similarly, anything that conditionally sets carry (an arithmetic operation) doesn't need to set the parity flag if the code is intended to run on a Z80. (This describes all CP/M-2.2 software I was interested in.) Another 16-byte table can therefore be used for arithmetic flag results. The 4-bit flag field of the 68000's status register is used as the index for this second table. The 68000 does have an overflow flag, so this is substituted for the parity bit of the 8080 (exactly as in the Z80). This causes one problem that is discussed in the Known Faults section. Treatment of the half carry bit is also discussed later because it doesn't fit into either of the tables.

The CP/M-80 environment simulation is greatly simplified (to my great disappointment) by the strong resemblance of CP/M-68K to CP/M-80. Most of the calls are directly translatable. There are a few exceptions, though, and they require the bulk of the code.

1. Any call referencing an FCB requires that the byte order of the Random Record field be switched, if the call uses that field.

2. CP/M-68K can't open a file in any but the base extent. You have to change such requests to an open in the base extent and then do a Random Read to the point you wanted.

3. Direct console I/O (BDOS #6) under CP/M-80 returns a null flag if no character is available. CP/M-68K waits for a character. A status check is performed first, and if a character is ready, it is fetched and returned to the simulation. Otherwise just a null status is returned.

4. Any call referencing an address (DMA or otherwise) needs to have that address translated to point into the 8080's code buffer. (This is a problem inherent to the simulation because the 8080 buffer cannot be placed in system memory at address 0.)

Some instruction simulations don't

do what you think they might. Specifically, the EI and DI instructions do not translate to the equivalent 68000 sequences. The object of using DI/EI pairs in an 8080 program is to prevent time-critical code from being interrupted or to prevent resource contention between interrupt routines and background processes that share resources. Because there are no 8080 "interrupts" possible under COM, there is nothing to block. (We won't even talk about simulating time-critical code!)

A few Z80 simulation routines are used in COM because of the overflow/parity flag problem discussed below. These are just an extension of the table approach used for the 8080—a large jump table and a bunch of subroutines. Extension to a full Z80 simulation is straightforward but would require a lot of code if the present jump table technique is kept.

Another approach to simulating instructions involves dividing the opcodes into classes, e.g., all MOVs handled by one subroutine that figures

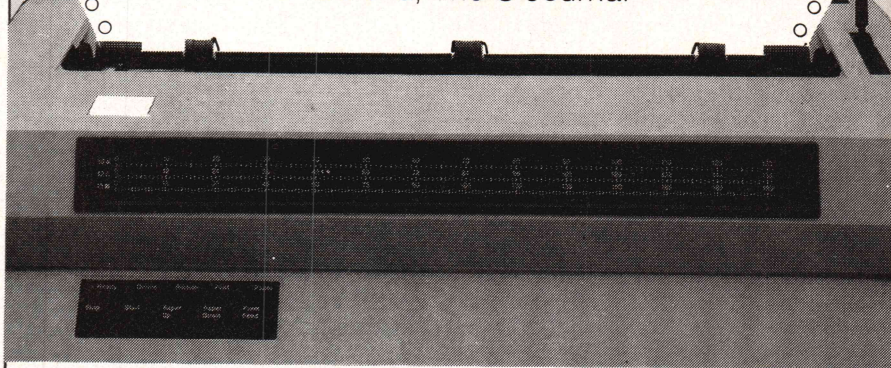
Programmer Essentials

"Offers many capabilities for a reasonable price"

W. Hunt, PC Tech Journal

"I highly recommend the C UTILITY LIBRARY"

D. Deloria, The C Journal



ESSENTIALS

200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for \$95.

\$100

THE C UTILITY LIBRARY

Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

\$185

ESSENTIAL GRAPHICS

Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save \$50 when purchased with above libraries. Available February, 1986.

\$250

Compatible with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

C Compiler Packages: Microsoft C - 319, Lattice or CI-C86 compilers - \$329. Save \$40 - \$50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add \$4 for UPS or \$7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, PO's.



ESSENTIAL SOFTWARE, INC

P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

Circle no. 138 on reader service card.

out what to move where by examining the opcode in further detail. Though this is much smaller code-wise than having the 63 similar sub-routines that I used, it also suffers a speed penalty that I just couldn't tolerate. I didn't buy a 68000 just to slow it down!

Known Faults

There are two problems with the simulation of the 8080's flags. The first is that the flags are more like those of the Z80 than the 8080 in that the parity flag reflects overflow status after arithmetic operations rather than parity. This fools some dynamically selected run-time packages such as the one used for BDS-C. There is minimal Z80 support in COM to handle the few extra instructions that BDS-C wants to use (LDIR and LDDR) so that programs compiled by it will run. (CPIR was also required by another program for the same reason.) This Z80 support could be extended as much as needed—up to and including full Z80 simulation.

The other flag problem is with the half carry bit. Simulating it would take a lot of overhead because there is no similar flag in the 68000. Therefore, assuming the only need for the flag is for the DAA instruction, this instruction is treated specially. Instructions that set the half carry bit meaningfully (ADDS, ADCs, and INR As) have additional code to store away the two operands and CY (if used) in special locations. The DAA simulation then recreates the HCY bit out of these stored values. A problem can arise when the flags are pushed and then pulled before the DAA is executed—an incorrect HCY is created if another addition-type operation occurs while the flags are supposedly saved. In practice, I only ran into this problem when using the 8080 DDT to trace through a DAA instruction. Be forewarned. If required, the simulation could be extended to eliminate this problem by proper simulation of the HCY bit, but this would slow the arithmetic routines down quite a bit.

BIOS calls to the disk drivers aren't allowed. I did this for safety reasons more than anything else—I didn't want possibly buggy simulations

playing with my hard disk without the protection of my BDOS. This limitation could easily be removed.

BDOS call #31 (Get DPH address) isn't supported. I didn't need this for any program I wanted to use so it was left out. Including it would involve getting the table from the 68000, copying it to somewhere where the "8080" could find it, and then returning a pointer to this copy of the table. Similarly, function #27 (Get ALLOC vector) isn't supported either. Using either of

these calls will cause an abort and an appropriate error message.

The IOBYTE and LOGIN vectors at 3 and 4 aren't supported.

Only one parsed FCB is supplied in the 8080's base page. The normally present second name at \$6C isn't parsed. (Note that CP/M-68K parses two full FCBs when COM is invoked. The first is the name of the .com program to run, and the second is used as this program's first FCB. CP/M-80's second "FCB" isn't one—it is only an

WARMST:	ORG 0	
	JMP BIOS	
	ORG 5	
	JMP BDOS	
BDOS:	ORG 0FF00H	
	HLT	; Service request.
	DB 0	; 0 is BDOS call, else BIOS.
	RET	
BIOS:	JMP WBOOT	
	JMP CONST	
	JMP CONIN	
	JMP CONOUT	
	JMP LIST	
	JMP PUNCH	
	JMP READER	
	JMP HOME	
	JMP SELDSK	
	JMP SETTRK	
	JMP SETSEC	
	JMP SETDMA	
	JMP READ	
	JMP WRITE	
	JMP LISTST	
	JMP SECTRN	
WBOOT:	HLT	; Service Request
	DB 1	; Passed to 68K BIOS (BIOS #1)
	RET	
CONST:	HLT	
	DB 2	; BIOS #2 (etc.)
	RET	
CONIN:	HLT	
	DB 3	
	RET	
CONOUT:	HLT	
	DB 4	
	RET	
LIST:	HLT	
	DB 5	
	RET	

Table 1

other name field in the first FCB. I didn't find anything that needed it, so I didn't go to the trouble of picking another name out of the 68K's command tail.)

An additional complication inherent to the simulation arises when you try to use programs running under COM to drive DMA devices. The hard disk controller in my system is a Morrow HD/DMA. In order for the 8080 simulation to be able to drive this controller, all addresses passed to the

board's DMA circuitry must undergo translation so that they point to the buffers in the "8080" program space, not the 68000's! Handling this tends to require a lot of code specific to each device supported, but it can be done. (COM was originally written for two reasons: to develop firmware for the 8085 used in my system's keyboard and to run the Morrow FORMATMW [hard disk formatter] program. Tracing this program pointed out the error in the HD/DMA documentation

that kept my own formatting program from working.) There is a conditional assembly flag in COM to include the support for the Morrow controller. You probably will never want this option, but I left the code in to serve as an example of extending COM to support a DMA device.

8080 Environment

The 8080 Environment is a 64K buffer, of which all but 512 bytes are available as TPA. This is probably the only real advantage of the simulation over real execution. The fake BIOS/BDOS (FDOS) starts at 8080 address \$FF00 and is in the form of a jump table followed by a service request table. The warmstart and BDOS jumps in the low page of the buffer point to these tables. There is no CCP because COM takes its place. The 8080 form of the FDOS is shown in Table 1 (page 78).

The service request handler performs a BIOS call to the 68000 if the parameter following HLT is not zero or a BDOS call if the parm is zero. In either case the appropriate parameters from the fake 8080 registers are translated (if required) and passed to the 68000 FDOS. The return values are then translated (if required) and stuffed into the 8080's pseudoregisters, and the simulation is continued.

Using COM

CP/M-80 programs are run by inserting the word COM in the normal command line. Examples of use are:

```
A>COM WS TEST.ASM
A>COM MAC TEST
A>COM LOAD TEST
A>COM DDT TEST.COM
A>COM LU -O JUNK -A TEST.COM
A>COM LDIR JUNK
A>COM MBASIC FFT.BAS
```

COM may be assembled with several optional trace facilities. Normally I create a separate version called COMT because the presence of the trace code slows down the simulation. A trace is specified by giving the full normal command line and then answering the prompts for the start and stop trace addresses. COMT will check each address before it simulates the opcode at that address for a match with either of the two limits and turn on or off the register dump appropriately. An example is shown

(Continued from page 78)

PUNCH:	HLT DB 6 RET	
READER:	HLT DB 7 RET	
HOME:	HLT DB 8 RET	; Normally blocked by ; the simulation.
SELDSK:	HLT DB 9 RET	; Ditto, etc.
SETTRK:	HLT DB 10 RET	
SETSEC:	HLT DB 11 RET	
SETDMA:	HLT DB 12 RET	
READ:	HLT DB 13 RET	
WRITE:	HLT DB 14 RET	
LISTST:	HLT DB 15 RET	; This one is allowed.
SECTRN:	HLT DB 16 RET END	

Table 1

AVAILABLE BACK ISSUES

1982	1983	1984	1985
No. 68—June	No. 77—March	No. 87—Jan.	No. 99—Jan.
No. 69—July	No. 78—April	No. 88—Feb.	No. 101—March
No. 70—Aug.	No. 79—May	No. 90—April	No. 102—April
No. 71—Sept.	No. 80—June	No. 91—May	No. 104—June
No. 72—Oct.	No. 81—July	No. 92—June	No. 105—July
No. 73—Nov.	No. 82—Aug.	No. 93—July	No. 106—Aug.
	No. 83—Sept.	No. 95—Sept.	No. 107—Sept.
	No. 84—Oct.	No. 96—Oct.	No. 108—Oct.
	No. 85—Nov.	No. 97—Nov.	No. 109—Nov.
	No. 86—Dec.		No. 110—Dec.

TO ORDER: send \$5 for 1 issue, \$4.50 each for 2-5, \$4 each for 6 or more to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto CA 94303

Name _____
Address _____
City _____ State _____ Zip _____ 3111F

**You read Dr. Dobb's Journal
And You Don't Subscribe?!**
Save over \$23.00 off newsstand prices for 2 yrs.
Save over \$10.00 for 1 yr.

Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.



Yes! Sign me up for
___ 2 yrs. \$47 ___ 1 yr. \$25

___ I enclose a check/money order
___ Charge my Visa, MasterCard,
American Express
___ Please bill me later

Name _____
Address _____

Zip _____
Credit Card _____ Exp. date _____
Account No. _____
Signature _____

Mail To: Dr. Dobb's Journal, PO Box 27809, San Diego, CA 92128

3049

8080 SIMULATOR
(Continued from page 79)

in Table 2 (page 81). The tracing addresses in this figure will trace every BDOS call made by the program.

The code for address prompting is rather stupid—it doesn't use line-buffered I/O. Because I almost never use tracing now that COM works, I didn't bother to fix this up. (The code was pulled out of my ROM monitor, where I didn't necessarily have any RAM for a buffer, which explains its strange structure.)

Other tracing code may be included in COMT. There is support for dumping (to the printer) FCB calls to the BDOS and for including a register dump at this time. All of these trace options were useful in debugging the simulation. You probably won't need them, but they illustrate one advantage of the simulation over the real thing—you can monitor events at any level of detail you want, provided you don't need real-time execution.

Teaching the Assembler Tricks (With a Hammer)

This section describes some of the tricks I used to make the ALCYON assembler (AS68) that is distributed with the CP/M-68K package do what I wanted when writing COM. Also described are some other tricks that I have used successfully in other assembly programs. (AS68 is also the assembler distributed with the Atari 520ST developer's package.)

At the beginning of Listing One are the register definitions used by COM. The form of these definitions was picked out of the AS68INIT file that you use the first time you run AS68. These definitions allow you to refer to the 68000 registers with more meaningful names than just "D3" and so on. The only real disadvantage of using names is that it is easy to forget which registers are in use and accidentally use one as a temporary that should have been saved first. Proper documentation helps in this. You also cannot use these new names in a "reg" directive.

One problem with these names (and all other "equ" defined symbols) is that you can't define them as global and use them in another file. The declarations must be repeated in each source file.

January 1986 #111

Expiration Date: April 30, 1986

Name _____ Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

Please circle one letter in each category:

I. My work is performed:

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

II. My primary job function:

- A. Software Project Mgmt/Spvr
- B. Hardware Project Mgmt/Spvr
- C. Computer Consultant
- D. Corporate Management
- E. Other

III. My company department performs:

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

IV. This inquiry is for:

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

V. Corporate Purchase Authority:

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about computers:

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

A reader service number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009	010
011	012	013	014	015	016	017	018	019	020
021	022	023	024	025	026	027	028	029	030
031	032	033	034	035	036	037	038	039	040
041	042	043	044	045	046	047	048	049	050
051	052	053	054	055	056	057	058	059	060
061	062	063	064	065	066	067	068	069	070
071	072	073	074	075	076	077	078	079	080
081	082	083	084	085	086	087	088	089	090
091	092	093	094	095	096	097	098	099	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230
231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290
291	292	293	294	295	296	297	298	299	999

Circle 999 to start a 12 month subscription at the price of \$29.97

Free!

Postage Paid!

Thank You!
Dr. Dobb's greatly appreciates your responses to questions I through VIII.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

Software Tools

P.O. Box 2157

Clinton, Iowa 52735-2157



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

Software Tools

P.O. Box 2157

Clinton, Iowa 52735-2157



Thank You!

**Dr. Dobb's greatly appreciates your
responses to questions I through VIII.**

Postage Paid!

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001 002 003 004 005 006 007 008 009
010 011 012 013 014 015 016 017 018
019 020 021 022 023 024 025 026 027
028 029 030 031 032 033 034 035 036
037 038 039 040 041 042 043 044 045
046 047 048 049 050 051 052 053 054
055 056 057 058 059 060 061 062 063
064 065 066 067 068 069 070 071 072
073 074 075 076 077 078 079 080 081
082 083 084 085 086 087 088 089 090
091 092 093 094 095 096 097 098 099
100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125 126
127 128 129 130 131 132 133 134 135
136 137 138 139 140 141 142 143 144
145 146 147 148 149 150 151 152 153
154 155 156 157 158 159 160 161 162
163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207
208 209 210 211 212 213 214 215 216
217 218 219 220 221 222 223 224 225
226 227 228 229 230 231 232 233 234
235 236 237 238 239 240 241 242 243
244 245 246 247 248 249 250 251 252
253 254 255 256 257 258 259 260 261
262 263 264 265 266 267 268 269 270
271 272 273 274 275 276 277 278 279
280 281 282 283 284 285 286 287 288
289 290 291 292 293 294 295 296 297
298 299 999

Circle 999 to start a 12 month subscription at the price of \$29.97

Dr. Dobb's Journal of Software Tools

January 1986 #111

Expiration Date: April 30, 1986

Name _____ Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

Please circle one letter in each category:

I. My work is performed:

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

II. My primary job function:

- A. Software Project Mgmt/Spvr
- B. Hardware Project Mgmt/Spvr
- C. Computer Consultant
- D. Corporate Management
- E. Other

III. My company department performs:

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

IV. This inquiry is for:

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

V. Corporate Purchase Authority:

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to ~~1,000~~ 999
- C. 100 to
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about computers:

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

**Product
Information**

Free!

Because of the conditional assemblies in COM, I needed to place some labels on lines by themselves, particularly MLOOP, the main looping point of the opcode simulation. Though the assembler allows this (provided you put a colon after the label), I was unable to make the label global using the ".globl" directive. However, if you do something like this:

```
.globl mloop
mloop:
~~ mloop: ; Why? I don't know!
```

you can get the declaration to work. This trick was found by examining the code produced by the C compiler. I believe that if the label is in uppercase you don't need this extra statement. Note that

```
.globl mloop
mloop: equ *
```

won't work because of the problem with "equ" described earlier.

The "offset" directive is extremely useful for generating data storage areas to be used with indexing. It would have been useful in COM, but it doesn't work. "Equ" must be used, and the programmer must count bytes in the storage areas to make sure nothing overlaps. Grrrrr. "Offset" wasn't accepted by the CP/M-68K Release 1.1 assembler. The Release 1.2 assembler works just fine—but neither linker will accept symbols defined in offset sections. Another "feature" to fix. Doesn't DRI test anything?

Another trick that may be useful in 68000 assembler programming is using the ".opd" directive to generate your own opcodes. These are particularly useful as stand-ins for less meaningful ".dc.w constant" sequences when generating data tables, although you may define instructions also. COM uses this trick to define the 68000's BDOS and BIOS trap instructions. Look at the file AS68INIT for more examples of its use. The only restriction for use is that the new opcode must follow the addressing rules of one of the existing 68000's instructions. Oh, for a true macro assembler. . . .

The C preprocessor, CP68, may be used as a poor man's macro assembler. If the assembly file (let's say TEST.MAC) looks something like this:

```
A>COMT TEST
Start trace at >5
Stop trace at >ff02
```

```
-AF- -BC- -DE- -HL- -SP- -S0- -S1- -S2- -S3- -PC- -op-
2300 4509 0200 0000 FFFE 0000 0000 0000 0000 0005 C3 JMP FF00

-AF- -BC- -DE- -HL- -SP- -S0- -S1- -S2- -S3- -PC- -op-
2300 4509 0200 0000 FFFE 0000 0000 0000 0000 FF00 76 HLT
Printed by BDOS #9
-AF- -BC- -DE- -HL- -SP- -S0- -S1- -S2- -S3- -PC- -op-
2300 4509 0200 0000 FFFE 0000 0000 0000 0000 FF02 C9 RET
A>
```

Table 2

DISCOVER THE LANGUAGE OF ARTIFICIAL INTELLIGENCE PROLOG V

Interpreter for MS-DOS/PC-DOS

At last! A Prolog with enough muscle to handle real-world applications for UNDER \$100! Discover why Japan has chosen Prolog as the vehicle for their "Fifth Generation Machine" project to design intelligent computers.

CHOOSE FROM TWO GREAT VERSIONS:

PROLOG V-Plus
\$99⁹⁵

- ☐ More Than 100 Predefined Predicates
- ☐ Large Memory Model (to 640K)
- ☐ Floating Point Arithmetic
- ☐ 150-Page User's Manual and Tutorial plus Advanced Programming Documentation
- ☐ Co-Resident Program Editor
- ☐ Calls to Co-Resident Programs
- ☐ Text and Graphic Screen Manipulation

PROLOG V
\$69⁹⁵

- ☐ 70 Predefined Predicates
- ☐ Small Memory Model
- ☐ Integer Arithmetic
- ☐ 122-Page User's Manual and Tutorial

FREE FREE FREE FREE FREE FREE
Programming in Prolog
by Clocksin & Mellish, \$17.95 value
Available with purchase of PROLOG
V-Plus only. Offer valid through
March 31, 1986.

STANDARD FEATURES ON BOTH:

- ☐ Clocksin & Mellish-Standard Edinburgh Syntax.
- ☐ Extensive Interactive Debugging Facilities
- ☐ Dynamic Memory Management (garbage collection)
- ☐ Custom-Designed Binder and Slipcase

THE CHOICE OF UNIVERSITIES

Generous university site licenses and an excellent teaching tutorial and reference guide have made PROLOG V the choice of universities nationwide. Call for details.

PHONE ORDERS: 1-800-621-0852 EXT 468

<input type="checkbox"/> PAYMENT ENCLOSED \$ _____		PROLOG V-Plus \$99.95
CA residents add 6% sales tax		PROLOG V 69.95
<input type="checkbox"/> CHARGE MY: <input type="checkbox"/> MasterCard <input type="checkbox"/> Visa		UPGRADE ONLY 40.00
Card No. _____ Exp. Date _____		Return factory diskette and \$30 plus \$10 Handling
Signature _____		SHIPPING:
Mr./Mrs./Ms. _____ (please print full name)		\$ 5.00 U.S.
Address _____		7.50 Canada
City/State/Zip _____		10.00 Caribbean, Hawaii Air
		20.00 Overseas Air
		COD Orders Not Accepted
		15 day check clearance

Upgrade to PROLOG V-Plus for only the difference in price plus a handling charge.

NO RISK OFFER
Examine the documentation at our risk for 30 days. If not fully satisfied, return with disk still sealed for full refund.



CHALCEDONY SOFTWARE
5580 LA JOLLA BLVD.
SUITE 126 D
LA JOLLA, CA 92037
(619) 483-8513

Circle no. 178 on reader service card.

How to go from UNIX to DOS without compromising your standards.

It's easy. Just get an industry standard file access method that works on both.

C-ISAM™ from RDS.

It's been the UNIX™ standard for years (used in more UNIX languages and programs than any other access method), and it's fast becoming the standard for DOS. Why?

Because of the way it works. Its B+ Tree indexing structure offers unlimited indexes. There's also automatic or manual record locking and optional transaction audit trails. Plus index compression to save disk space and cut access times.

How can we be so sure C-ISAM works so well?

We use it ourselves. It's a part of INFORMIX®, INFORMIX-SQL and File-it!™, our best selling database management programs.

For an information packet, call (415) 322-4100. Or write RDS, 4100 Bohannon Drive, Menlo Park, CA 94025.

You'll see why anything less than C-ISAM is just a compromise.



RELATIONAL DATABASE SYSTEMS, INC.

© 1985, Relational Database Systems, Inc. UNIX is a trademark of AT&T. INFORMIX is a registered trademark and RDS, C-ISAM and File-it! are trademarks of Relational Database Systems, Inc.

Circle **no. 170** on reader service card.

8080 SIMULATOR
(Continued from page 81)

```
#include "MACRO.H"
label    testmac(1,2,3)
end
```

and the file MACRO.H looks like this:

* File produced by using CP68 as a macro
* assembly preprocessor on a .MAC file.

```
#define testmac(x,y,z) dc.w x \
    .dc.w y \
    .dc.w z
```

and if the files are processed like this:

```
A>CP68 TEST.MAC TEST.S ; CP/M 1.1
```

or

```
A>CP68 -P TEST.MAC TEST.S ; CP/M 1.2
```

then the output file (TEST.S) will look like this:

* File produced by using CP68 as a macro
* assembly preprocessor on a .MAC file.

```
<cr>
<cr>
<cr>
label    .dc.w 1
        .dc.w 2
        .dc.w 3
end
```

The <cr> lines are additional blank lines, one for each of the lines of a #define in the .H file.

Notes

The principal reference for the 8080 model was the *MCS-80/85 Family User's Manual* by Intel Corporation.

This program is released to the public domain with the stipulation that it be used for noncommercial purposes and that appropriate credit be given in any upgrade.

DDJ

(Listings begin on page 104)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 5.**

Disclone Service Won't Make You Nervous

Technical support, personal service, competitive prices.

Disclone full service quality tested diskette duplication, packaging, documentation production and processing ensures precise duplication, thorough quality control, and expedient response to your requirements.

NOclone state of the art hardware based copy protection is true piracy protection for authorized allotments only. Each application is uniquely encrypted. Install routines are coded for nontransferrable hard disk allotments.

Commitment dates are guaranteed. Fast turnover

- up to 1000 in 24 hours, any format.
- up to 10,000 in one week, any format.

disclone
92909E

DISCLONE SOFTWARE PRODUCTION SERVICES

1585 North Fourth Street, San Jose, California 95112
(408) 947-1161 OUTSIDE CA: 1-800-826-4296

Circle no. 204 on reader service card.

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K& RC. Source code included, No Royalties, Powerful & Portable.

BTree Library

75.00

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver

40.00

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make

59.00

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

For more information call or write:

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J6
(416) 825-0903
(416) 844-2610

softfocus

Credit cards accepted.

Dealer inquiries invited.

WINDOWS FOR DATA™

Featuring One-Step Data Entry♦

Now you can code fast, powerful data entry windows, improve user convenience - reduce input errors.

All the power, convenience and flexibility of the #1 window utility for the IBM PC. Our WINDOWS FOR C™ combined with a professional window-based data entry system.

Complete control over screen display and entry of data within a convenient flexible window environment.

WINDOWS FOR C WINDOWS FOR DATA
(Includes WINDOWS FOR C)

PCDOS	\$ 195	\$ 295
PC/XENIX	\$ 395	\$ 595
UNIX	CALL	CALL

WINDOWS FOR DATA™ provides versatile, easy-to-use data entry functions that operate within windows.

CAPABILITIES INCLUDE:

- Pop-up data entry windows
- Multiple field types
- Data validation functions
- Field-specific & context-sensitive help
- Lotus-style menu design
- Single field entry option
- Date, time and string utilities
- Dynamic control of data-entry environment

- ♦ User input to data-structure variables without intervening code.



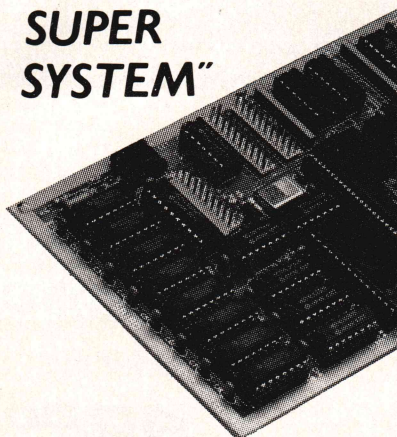
**Vermont
Creative
Software**

21 Elm Ave.
Richford, VT 05476
802-848-7738, ext. 31

Full source available. Master Card & Visa accepted. Shipping \$ 3.50. VT residents add 4% tax.

Byte Magazine called it.

"CIARCIA'S SUPER SYSTEM"



The SB180 Computer/Controller

Featured on the cover of Byte, Sept. 1985,
the SB180 lets CP/M users upgrade to a
fast, 4" x 7 1/2" single board system.

- 6MHz 64180 CPU
(Z80 instruction superset), 256K RAM,
8K Monitor ROM with device test, disk
format, read/write.
- Mini/Micro Floppy Controller
(1-4 drives, Single/Double Density,
1-2 sided, 40/77/80 track 3 1/2", 5 1/4"
and 8" drives).
- Measures 4" x 7 1/2", with mounting holes
- One Centronics Printer Port
- Two RS232C Serial Ports
(75-19,200 baud with console port
auto-baud rate select).
- Power Supply Requirements
+5V +/-5% @500 mA
+12V +/-20% @40mA
- ZCPR3 (CP/M 2.2/3 compatible)
- Multiple disk formats supported
- Menu-based system customization

SB180-1

SB180 computer board w/256K
bytes RAM and ROM monitor
.....\$369.00

SB180-1-20

same as above w/ZCPR3, ZRDOS
and BIOS source.....\$499.00

-Quantity discounts available-

NEW

COMM180-M-S

optional peripheral board adds
1200 bps modem and SCSI
hard disk interface.

TO ORDER
CALL TOLL FREE
1-800-635-3355

TELEX
643331

For technical assistance or
to request a data sheet, call:

1-203-871-6170



Micromint, Inc.
25 Terrace Drive
Vernon, CT 06066

C CHEST LISTING

(Text begins on page 18)

```
1 #include <stdio.h>
2 #include <dir.h>
3 #include <process.h>
4 #include <errno.h>
5 #include <fcntl.h>
6 #include <signal.h>
7 #include <dos.h>
8
9 /* SH.C: A shell for MSDOS.
10 *
11 * Copyright (C) 1985 Allen I. Holub. All rights reserved.
12 *
13 *
14 *
15 * This file contains an MSDOS shell that operates in conjunction
16 * with command.com. It has several built-in commands (see below).
17 * It recognizes several semi-colon separated commands on a line
18 * & does wild card expansion. It can take commands interactively
19 * or from the command line (but command.com will intercept the
20 * re-direction if you have any on the command line). It supports
21 * simple batch files and will expand several $ arguments.
22 *
23 * It does not support redirection yet.
24 *
25 *
26 * Usage: sh [-cvx] <args>
27 *
28 * Invocation:
29 *
30 * sh
31 * sh -i          shell entered in interactive mode. If -i is given
32 *                any command line arguments that follow will be
33 *                assigned to $0 $1 etc. $0 will point at the
34 *                leftmost argument following the -i.
35 *
36 * sh -c <string> commands are read from string. If several strings
37 *                are present they are concatenated together before
38 *                execution. The 'c' may be upper or lower case.
39 *
40 * sh <file> args... commands are taken from <file>. Args are expanded
41 *                to correspond to $0 $1 etc inside the file. For
42 *                DOS compatibility %0 %1 etc are also recognized.
43 *                $0 will be <file> itself.
44 *
45 * sh -q          Strip quotes from quoted argument strings. Usually
46 *                they're left in so that a spawned process can
47 *                assemble its argv correctly.
48 * sh -v          Print input lines to the shell as they are read
49 *                (same as "set verbose=1").
50 * sh -x          Print lines as they are executed. (same as
51 *                saying "set echo=1" in the shrc.bat file.
52 *
53 *
54 * Environment variables:
55 * * CMDLINE (set) Holds the complete, 2048 byte, command line
56 *                that can't be passed via MSDOS.
57 * * PROMPT (used) Defines the prompt string. Any character
58 *                or $arg may be used. Default prompt is
59 *                [$s:$p].
60 * * SHLEV (set) Current shell level (0 is outermost). Remember
61 *                that batch files are executed in their own
62 *                shell.
63 * * SWITCHAR (used) Use first character in line to designate
64 *                command line switches.
65 *
66 * Files:
67 * * /shrc.bat Executed every time a shell is created.
68 * * /login.bat Executed when a level 0 shell is created.
69 * * /logout.bat Executed when "logout" command is executed
70 *
71 * Built in commands:
72 * * alias [name [wordlist]]
73 * * cd <directory name>
74 * * exit
75 * * history
76 * * logout
77 * * pwd
78 * * rem
79 * * setenv <name> [=] <value>
80 * * set [[cmd|echo|verbose|$arg] [= val]]
81 * * shift
82 * * unalias <name>
83 * * unset <name>
84 * * ! !! !<num> !<pat>
85 * * ^ ^^ ^<num> ^<pat>
86 * * |> [name]
87 * * |< [name]
88 * * #
89 *
90 * Pre-defined shell variables (may use either % or $) :
91 *
92 * * $<num> one argument in batch file. $0 is file name.
93 * * $* All the $<num>s concatenated with spaces between them.
94 * * $p Full path name of current directory.
95 * * $! Current History number.
96 * * $$ Nesting level of current shell. 0 is the outermost
97 *
98 *
99 * Special characters:
100 * * ; Used to delimit two commands on one line.
101 * * *? Name containing these is expanded to matching directory
102 * * entry.
103 *
104 * Special characters aren't recognized in quoted strings or
105 * when preceded by a backslash.
106 * All lines with # in the left-most column are ignored.
```



```

107 *-----
108 */
109
110 extern int      access (char*, int      ); /* in stdio library */
111 extern int      bdos   (int,   int, int );
112 extern int      chdir  (char*   );
113 extern int      errno;
114 extern FILE     *fopen  (char*, char*   );
115 extern int      fseek  (FILE*, long, int );
116 extern long     ftell   (FILE*   );
117 extern char     *getenv (char*   );
118 extern char     *getcwd (char*, int   );
119 extern char     *malloc (unsigned );
120 extern int      putenv  (char*   );
121 extern int      ( *signal (int,int(*) ) )();
122 extern int      strlen  (char*   );
123 extern char     *strcpy (char*, char* );
124
125
126 extern char     *copy   (char*, char*   ); /* source is in: */
127 extern void     del_dir (DIRECTORY * ); /* /src/tools/cpy.c */
128 extern void     dir     (char*, DIRECTORY* ); /* /src/tools/dir.c */
129 extern char     *efgets (char*, int, FILE* ); /* /src/tools/efgets.c */
130 extern DIRECTORY *mk_dir (int      ); /* /src/tools/dir.c */
131 extern char     *next    (char**, int, int ); /* /src/tools/next.c */
132 extern char     *skipto  (int,   char*, int ); /* /src/tools/skipto.c */
133 extern char     *strsave (char*   ); /* /src/tools/strsave.c */
134 extern int      unargv  (int,char**,char*,int,int); /* /src/tools/unargv.c */
135
136 extern void     unsetvar( char* ); /* ./var.c: */
137 extern int      setvar( char*, char* );
138 extern void     printalias();
139 extern void     printvars();
140 extern int      getvar(char**, char**, int*);
141
142 extern void     print_hist(FILE*); /* ./hist.c */
143 extern int      get_hnum();
144 extern void     history( char*, int);
145
146 /*-----
147
148 #ifdef DEBUG
149     static int      Lev = -1;
150     # define TRACE(p) printf("%s{ entering %s\n", ++Lev * 4, "", p)
151     # define END_TRACE(p) printf("%s} exiting %s\n", Lev-- * 4, "", p)
152     # define DIAG(f,a) printf(f,a)
153 #else
154     # define TRACE(p)
155     # define END_TRACE(p)
156     # define DIAG(f,a)
157 #endif
158
159
160 #ifdef STR_CMDS
161     # define PSTR(subr,str) printf("%s < %s>\n", subr, str);
162 #else
163     # define PSTR(subr,str)
164 #endif
165
166 /*-----
167
168 #define VER      "1.0" /* Version number */
169
170 #define DOSMAXLINE 127 /* Maximum line number permitted by DOS */
171 #define MAXLINE    (2048+1) /* Largest input command line in bytes +1 */
172 #define MAXDIR     128 /* Largest number of objects on cmd line */
173 #define CNTL_Z     ('Z'-'')
174 #define COMMENT    '#' /* Deliniates comments */
175
176 #define ISQUOTE(c) ((c)=='"' || (c)=='\'')
177 #define ISWHITE(c) ((c)==' ' || (c)=='\t')
178 #define SKIPWHITE(p) while( ISWHITE(*p) ){ p++; }
179 #define ISVAR(c) ((c)=='$' || (c)=='%')
180
181
182 /* Possible modes in which shell can operate */
183 #define FILEMODE 0 /* Get input from a file */
184 #define INTERACTIVE 1 /* Get input interactively from stdin */
185 #define COMMAND 2 /* Get input from the command line */
186
187 #define PMODE() ( Mode==COMMAND ? "COMMAND" : \
188                 (Mode==FILEMODE ? "FILE" : "INTERACTIVE") )
189
190 /*-----
191 * Token definitions for built-in commands.
192 */
193
194 typedef enum
195 {
196     ALIAS,
197     CD,
198     CMD,
199     EXIT,
200     HISTORY,
201     LOGOUT,
202     PWD,
203     REM,
204     SET,
205     SETENV,
206     SHIFT,
207     UNALIAS,
208     UNSET
209 } TOKEN;
210
211 /*-----
212 * Global variables:
213 */
214
215 static char** Numv ; /* Vector array for expanding $<num> vars */
216 static int Numc = 0; /* count of valid entries in the above */
217 static char Ibuf[MAXLINE]; /* Input buffer */

```

(Continued on next page)

TURBO *Power*™ UTILITIES

FOR

Turbo Pascal™ PROGRAMMERS

**Improve Code Performance
Find Subtle Bugs
Automate Tedious Tasks**

Supports Turbo Pascal 2.0 & 3.0
IBM PC/XT/AT & True Compatibles
PCDOS 2.X & 3.X
192K RAM DSDD Drive

**If You Really Use Your
Pascal Compiler You Need
These Tools!**

- Pretty Printer
- Pascal Structure Analyzer
- Execution Profiler
- Execution Timer

**Advanced Text Processing
& Command Automation**

- Pattern Replacer
- Difference Finder
- Command Builder
- File Finder
- Super Directory

**Where Else Can You Get
500K of Integrated, Useful,
Tested, Fully Documented
Source Code for \$95?**

- 140 Page Printed User Manual
- Quick Reference Card
- Detailed Programmer's Manual on Disk
- Complete Turbo Pascal Source Code
- 6 Bonus Utilities with Source!
- Tax & U.S. Postage Included
- Executable only version \$55

MC/Visa Orders TOLL FREE

(USA) 800-538-8157 x830

(CAL) 800-672-3470 x830

**Brochures, Questions, PO's
call 408-378-3672**

Checks or Money Orders

TurboPower Software

**478 W. Hamilton Ave., Suite 196
Campbell, CA 95008 U.S.A.**

INTERNATIONAL REPRESENTATIVES —
Switzerland: Software Haus 064-512651
Japan: Southern Pacific Ltd 045-314-9514
England: The Core Store 0606-45420
Canada: Software Commodities 416-865-1600
Holland: SCOS PC-Center 020-106922
Norway: Polysoft 03-82575

Turbo Pascal is a Trademark of Borland International

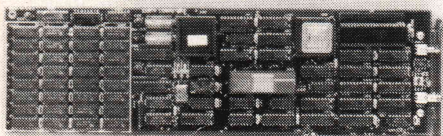
Circle **no. 207** on reader service card.

LOOKING FOR AT PREFORMANCE FROM YOUR PC?



**EARTH HAS IT FOR
LESS THAN \$1,000!**

YOUR SEARCH IS OVER!! EARTH COMPUTERS' exciting new high-speed, 80286 accelerator card, **TurboACCEL-286™**, is just what you've been looking for. The **TurboACCEL-286** will boost your PC performance up to **Five times...** its completely software transparent...and its only **\$995!** **TurboACCEL-286** will function with most operating systems and application programs (unlike other so-called accelerator boards).



The **TurboACCEL-286** features a high-speed, 8MHz, 80286 processor, 512Kbytes of RAM (expandable to 1Mbytes), a switch for 8088 operation, and facilities for an 80287 math co-processor. It occupies one expansion slot, is completely compatible with most PCs and is software transparent. End your search for AT performance. Order the **TurboACCEL-286** today! Call or write:

EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728
TELEX: 910 997 6120 EARTH FV

(714) 964-5784

Ask about EARTH COMPUTERS' other fine PC and S-100 compatible products.

Circle no. 179 on reader service card.

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

218 static int Cmd - 1; /* Generate CMDLINE env with spawned proc */
219 static int Switchar - '-'; /* Designates command line switches */
220 static int Verbose - 0; /* Print input lines as they're read -v */
221 static int Echo - 0; /* Print commands as they're executed -x */
222 static int Noquotes - 0; /* Strip " or ' from quoted args -q */
223 static int Shlev - -1; /* Nesting level of the current shell */
224 static char *Filename - 0; /* Full path name of file specified in */
225 /* Filemode input. */
226
227 /*-----
228 * Set up input mode for file mode processing. Note that Last_posn and
229 * Ateof are used by file_input().
230 */
231
232 char *file_input();
233 static int Mode - FILEMODE;
234 static char *(*Ifunct)() - file_input;
235 static long Last_posn - 0L;
236 static int Ateof - 0;
237
238 reset_fileinput()
239 {
240     Last_posn = 0L;
241     Ateof = 0;
242     Mode = FILEMODE;
243     Ifunct = file_input;
244 }
245
246 /*-----
247 * Making isdigit into a subroutine makes processing marginally
248 * easier in exp_vars (below).
249 */
250
251 digit(c)
252 {
253     return( '0' <= c && c <= '9' );
254 }
255
256 /*-----
257 * Input functions: interactive input() Gets input from keyboard
258 * command input() Gets input from cmd line
259 * file_input() Gets input from a file.
260 *
261 * Only one of the three will be used depending on the way that
262 * the shell was invoked. All three return 0 on end of input, a
263 * pointer to the beginning of the current input line on success.
264 * The input line will have been loaded into the global array
265 * Ibuf, which is assumed to be dimensioned to MAXLINE characters.
266 */
267 /*-----
268
269 char *interactive_input()
270 {
271     register char *rval = NULL;
272     TRACE("interactive_input");
273
274     *Ibuf = 0;
275     if( efgets(Ibuf, MAXLINE, stdin) )
276         rval = Ibuf;
277
278     END_TRACE("interactive_input");
279     return rval;
280 }
281
282 /*-----
283
284 char *command_input()
285 {
286     static int have_been_called = 0;
287     register char *rval = NULL;
288
289     TRACE("command_input");
290
291     if( !have_been_called )
292     {
293         unargv( Numc, Numv, Ibuf, MAXLINE, ' ' );
294         rval = Ibuf;
295         have_been_called++;
296     }
297
298     END_TRACE("command_input");
299     return rval;
300 }
301
302 /*-----
303
304 char *file_input()
305 {
306     /* Get input for file mode. This kludge is required because
307     * a child process will close any open files when it exits.
308     * Consequently we open the file, get a line, remember
309     * the position within the file, and then close the file
310     * on each call. On the next call we'll return to the
311     * position we remembered in the previous call.
312     */
313
314     register char *rval = NULL;
315     register FILE *fp;
316
317     TRACE("file_input");
318
319     if( !Ateof )
320     {
321         if( !(fp = fopen(Filename, "r")) )
322             fprintf(stderr, "Sh: Can't open batch file < %s>\n",
323                     Filename);
324     }

```



```

325         else if( !fseek(fp, Last_posn, 0) )
326         {
327             /* Get a line from the buffer. Note that if the
328              * file doesn't have it's last line terminated
329              * with a carriage return, efgets will return
330              * true even though we're at end of file
331              * thus the call to feof.
332              */
333
334             *Ibuf = 0;
335             rval = efgets(Ibuf, MAXLINE, fp) ? Ibuf : NULL ;
336             if( !(Ateof = feof(fp)) )
337                 Last_posn = ftell( fp ) - 1;
338             fclose( fp );
339         }
340     }
341
342     END_TRACE( "file_input" );
343     return( rval );
344 }
345
346 /-----*/
347
348 int  has_wild( bp )
349 register char  *bp;
350 {
351     /*      Return true if the string has a * or ? in it
352     */
353
354     for( *bp ; ++bp )
355         if( *bp == '*' || *bp == '?' )
356             return 1;
357
358     return 0;
359 }
360
361 /-----*/
362
363 int  strip( src )
364 register char  *src;
365 {
366     /*      Take care of special characters in a string (* and ?).
367     *
368     *      Copy src onto itself, stripping out backslashes. If a
369     *      * or ? which isn't preceeded by a backslash is found
370     *      return 1, else return 0. If the first character in the
371     *      string is a quote then * and ? aren't special.
372     */
373
374     register char  *dest = src;
375     int            special = 0;
376
377     TRACE("strip");
378
379     while( *src )
380     {
381         if( *src == '\\' )
382         {
383             /* Copy the char. following the \ into
384              * dest and then, if we aren't at end of
385              * string, advance src to point past the
386              * escaped character
387              */
388
389             *dest++ = ++src;
390             if( *src )
391                 ++src;
392         }
393         else if( ISQUOTE(*src) )
394         {
395             /* Copy a quoted string verbatim, removing
396              * the quotes if Noquotes (-q) was given on
397              * the command line.
398              */
399
400             if( Noquotes )
401                 ++src;
402
403             while( *src && !ISQUOTE(*src) )
404             {
405                 if( src[0] == '\\' && src[1] )
406                     *dest++ = *src++;
407
408                 *dest++ = *src++;
409             }
410
411             if( *src ) /* Then *src is a quote */
412             {
413                 if( Noquotes )
414                     src++;
415                 else
416                     *dest++ = *src++;
417             }
418         }
419         else
420         {
421             /* Just do the copy. Set special to true
422              * if we copy a special character.
423              */
424
425             if( *src == '*' || *src == '?' )
426                 special = 1;
427
428             *dest++ = *src++;
429         }
430     }
431
432     *dest = '\0';
433
434     END_TRACE("strip");

```

(Continued on next page)

Put More UNIX™ in Your C.

Unitools \$99 MAKE, DIFF and GREP

These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them. GREP can search one or many files looking for one pattern or a host of patterns.



"Z" \$99

A Powerful "vi"-type Editor: Similar to the Berkeley "vi" editor, "Z"s commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include "undo," sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.



PC-LINT \$99

Error Checking Utility

A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn't expect at this low price.



SunScreen \$99

Low-priced Screen Utility

This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is \$199.

Compatible with all leading MS/
PC-DOS C compilers.

SPECIAL OFFER:
Unitools, "Z,"
PC-LINT and Sun-
Screen All for only

\$349

To order or for information call:

TECWARE
1-800-TEC-WARE

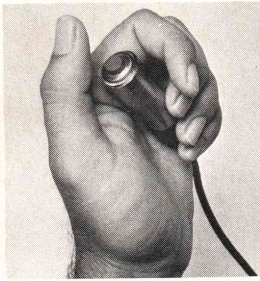
(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories. MANX AZTEC TM Mass Software Systems, Inc. PC LINT TM GIMPLE software. SunScreen TM SunTec. MS-DOS TM Microsoft.

Circle no. 223 on reader service card.

Introducing Periscope II Professional Debugger and Break-out Switch



New Peri-
scope II
includes
a remote
break-out
switch that
does not
need its
own slot!

(Periscope is) "the best value in development tools currently on the market . . . the most essential element of my 'developer's toolbox'."

—Jeff Garbers

"Very powerful for debugging and testing . . . Better than Atron by far."

—Wynn Bailey

The break-out switch "really sets Periscope apart from the typical software-only debuggers." Hung system or locked keyboard? Press the switch to get control!

Periscope's symbol support "beats the day-lights out of snooping through a map file and making notes". See high-level line numbers and source code, too!

"Feel right at home" in no time with commands that logically extend Debug's!

Periscope's speed makes other debuggers "look absolutely sluggish"! It's written entirely in assembler and uses DOS only for file access.

Has all the standard features plus:

- Debug with over 75 breakpoint options
- New! Write your own breakpoint tests
- New! Traceback
- New! Do in-line symbolic assembly
- Debug using one or two monitors
- Recall command lines
- New! Debug with high-level source code
- New! Redefine windows while debugging
- New! View text files while debugging
- Debug device drivers, non-DOS and memory-resident programs
- New! Customize Periscope via user exits
- New! Display 8087/80287 status
- New! Use Periscope with an EGA

Periscope requires: IBM PC, XT, AT, or close compatible; DOS 2.0 & later; 128K RAM; 1 Disk Drive; 80-column Monitor.

Periscope II, break-out switch, manual, reference card and software . . . \$145!

Periscope I also includes the write-protected RAM board to protect crucial debugger code. It's just \$295!

The US Navy gets Periscopes from us . . . shouldn't you? Order today!

Order/Information Call Toll-Free:



800-722-7006



30-Day Money-Back Guarantee

Data Base Decisions • 404/256-3860
14 Bonnie Lane • Atlanta, GA 30328
Circle no. 130 on reader service card.

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

435
436         return( special );
437     }
438
439     /*-----*/
440
441     char *nextarg( lp )
442     char **lp;
443     {
444         /*      Get the next, space delimited, argument from the string
445          *      pointed to by *lp. Return a pointer to the argument and
446          *      update *lp to point past it. Leading white space is
447          *      skipped.
448          */
449
450         register char *start, *line = *lp;
451
452         TRACE("nextarg");
453         SKIPWHITE( line );
454
455         if( !*line )
456             start = (char *) 0 ;
457         else
458             {
459                 start = line++;
460
461                 line = skipto( ISQUOTE(*start) ? *start : ' ', line , '\\');
462
463                 if( ISQUOTE(*line) )
464                     line++;
465
466                 if( *line )
467                     *line++ = '\\0' ;
468
469                 *lp = line;
470             }
471
472         END_TRACE("nextarg");
473
474         return start;
475     }
476
477     /*-----*/
478
479     int exp_dir( buf, maxcount )
480     char *buf;
481     {
482         /*      Remake buf, expanding any wild card characters into
483          *      their proper names. That is, if buf containing the string:
484          *      "foo * bar" and the current directory contains the
485          *      files A, B and C then on exit, buf will point
486          *      at the string "foo a b c bar". Expanded entries are
487          *      sorted. Return 0 if a wild card that couldn't be
488          *      expanded was found, else return 1.
489          */
490
491         register DIRECTORY *dp = 0 ;
492         register char *arg, *sbuf, *p ;
493         int i, rval = 1;
494
495         TRACE("exp_dir");
496
497         if( !(dp = mk_dir(MAXDIR)) )
498             goto abort;
499
500         dp->files = 1;          /* Get all files */
501         dp->dirls = 1;          /* and all directories */
502         dp->path = 1;          /* and prepend the path name if given */
503         dp->sort = 1;          /* the list should be sorted */
504
505         /*      Get the arguments from the input src, one at a time,
506          *      putting them into the dirv array of a DIRECTORY structure
507          *      (one argument per dirv entry).
508          *
509          *      If the argument has special characters (* or ? not
510          *      preceded by a \ or enclosed by quotes) then expand to a
511          *      directory using dir(), otherwise just put the argument
512          *      into the dirv array directly.
513          *
514          *      Buf will grow larger if anything is expanded but it
515          *      won't get larger than maxcount.
516          */
517
518         for( sbuf = buf; (arg = nextarg(&sbuf)) && dp->maxdirs > 0; )
519         {
520             if( strip(arg) )
521             {
522                 i = dp->maxdirs;
523
524                 dir( arg, dp );
525
526                 if( dp->maxdirs == i )
527                 {
528                     /*      dir() didn't do anything */
529
530                     fprintf(stderr, "Sh: Can't expand <%s>\n", arg);
531                     rval = 0;
532                     goto abort;
533                 }
534             }
535             else
536             {
537                 /*      Add a command to dirv. First malloc space
538                  *      for it. Then copy it the arg into the
539                  *      malloc'ed space & put it into dirv.
540                  *      We use malloc in order to make it easier
541                  *      to free the space used by the DIRECTORY

```



```

542      * structure.
543      */
544      if (! ( p = malloc( strlen(arg)+1 ) ))
545      {
546          fprintf(stderr, "Sh: out of memory !\n");
547          goto abort;
548      }
549      strcpy( p, arg );
550
551      *( dp->lastdir )++ = p ;
552      --( dp->maxdirs );
553      ++( dp->nfiles );
554      }
555
556      }
557
558      i = unargv(dp->nfiles+dp->ndirs, (char **)dp->dirv,sbuf,maxcount, ' ');
559
560      if( dp->maxdirs <= 0 || i >= maxcount-1 )
561          fprintf(stderr, "Sh: command line too large, truncating\n");
562
563      abort: if( dp )
564          del_dir( dp );
565
566      END_TRACE("exp_dir");
567      return( rval );
568  }
569
570  /*-----*/
571
572  char *search( fname, ext )
573  char *fname, *ext;
574  {
575      /* Search for fname.ext in the current PATH. Return a pointer
576       * to the full path name if you find it (0 if you don't).
577       */
578
579      static char  pathname[80], pbuf[129];
580      register char *p;
581      char *paths;
582
583      /* Assemble the pathname by concatenating fname and ext
584       */
585      TRACE("search");
586
587      if( strpbrk(fname, ".") ) /* If file name already has an */
588          ext = ""; /* extension don't add another. */
589
590      sprintf( pathname, "%0.32s.%0.3s", fname, ext );
591
592      if( access(pathname, 04) < 0 )
593      {
594          /* The file doesn't exist in the current directory.
595           * If fname contains the characters \ or / or if
596           * the PATH environment isn't set, return a NULL,
597           * else search for it along the path. strpbrk() is
598           * a microsoft and Lattice library function. It'll
599           * return true if fname contains a / or a \.
600           */
601
602          if( strpbrk(fname, "\\ /") || !(p = getenv("PATH")) )
603              *pathname = '\0';
604          else
605          {
606              strncpy( (paths = pbuf), p, 129 );
607
608              while( p = next( &paths, ' ', -1 ) )
609              {
610                  sprintf(pathname, "%0.50s\\%0.20s.%0.3s",
611                          p, fname, ext);
612
613                  if( access( pathname, 04 ) >= 0 )
614                      break;
615                  else
616                      *pathname = '\0';
617              }
618          }
619      }
620
621      END_TRACE("search");
622
623      return( *pathname ? pathname : NULL );
624  }
625
626  /*-----*/
627
628  int execute( buf )
629  char *buf;
630  {
631      /* Execute a command. Return the programs exit status or
632       * -1 if the program didn't execute. This routine assumes
633       * that buf is at least DOSMAXLINE characters long.
634       */
635
636      register int  rval = 0;
637      register char *name;
638      static char  envstr[MAXLINE+8] = "CMDLINE=";
639
640      TRACE("execute");
641      PSTRT( "execute(1) buf=", buf );
642
643      /* Create the CMDLINE environment variable if Cmd it true
644       * (it can be set false with a "set cmd=0." If cmd is false
645       * then generate a "CMDLINE=" with no argument. This is
646       * necessary because MSDOS doesn't support a unset command.
647       */
648
649      if( Cmd )

```

(Continued on next page)

Work Smart with These Powerful C Utilities

Get more value from your C system. Boost program quality and slash development time with these professional utilities for leading C-compiler systems.

C Utility Library ~~\$185~~ \$155 Over 300 C subroutines

C and assembler source code and demonstration programs for screen handling, color printing, graphics, DOS disk and file functions, memory management and peripherals control.

C-tree ~~\$395~~ \$329 B-Tree database system

Store, update and retrieve records easily. High-level multi-key ISAM routines and low-level B-Tree functions. Available for MS-DOS, CP/M-86, and CP/M-80. Easily transported. Adaptable for network and multiuser. Includes source.

PHACT ~~\$295~~ \$200 Data Base Record Manager

Includes high-level features found in larger database systems. Available for MS-DOS, CP/M-86 and CP/M-80.

Pre-C ~~\$395~~ \$329 LINT-like source code analyzer

Locates structural and usage errors. Cross-checks multiple files for bad parameter declarations and other interface errors.

Windows for C ~~\$195~~ \$165 Versatile window utility

Supports IBM PC compatible and some non-compatible environments.

PANEL ~~\$295~~ \$235 Screen generating utility

Create custom screens via simple, powerful editing commands. Select colors, sizes and types, edit fields. Includes direct input utility.

HALO ~~\$280~~ \$199 Ultimate C graphics

A comprehensive package of graphics subroutines for C. Supports multiple graphics cards.

PLINK-86 ~~\$395~~ \$315 Overlay linker

Includes linkage editor, overlay management, a library manager and memory mapping. Works with Microsoft and Intel object format.

To order or for information call:

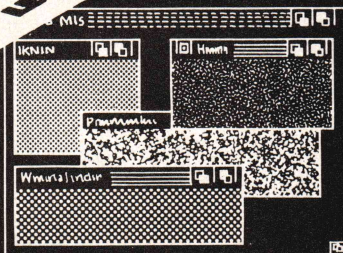
TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories, C-tree TM Phoenix, Inc. PHACT TM PHACT ASSOC. Pre-C TM PLINK-86 TM PHOENIX, HALO TM Media Cybernetics, Inc. PC-INT TM GIMPLE software. PANEL TM Randall Computer Systems, Ltd. WINDOWS FOR C TM Creative Solutions, C.F.M. TM DRI.

Circle no. 222 on reader service card.

NEW FOR
AMIGA™



MetaScope: The Debugger

MetaScope gives you everything you've always wanted in a debugger:

- **Multiple Windows**
Open and close, move through memory, display data or disassembled code.
- **Full Symbolic Capability**
Read symbols from files, define new ones, use anywhere.
- **Powerful Expression Evaluation**
Use any standard assembler operators or number formats.
- **Direct to Memory Assembler**
Enter instruction statements for direct conversion to code in memory.
- **and More!**
Log file for operations and displays, breakpoint and trace execution, modify/search/fill memory, etc.

MetaScope is designed to fully utilize the capabilities of the Amiga in debugging your programs. If you're programming the Amiga,™ you can't afford to be without it.

\$95 (California residents + 6%).
Visa/MasterCharge accepted.

Amiga is a trademark of Commodore-Amiga Inc.

Metadigm, Inc.

19762 MacArthur Blvd.
Suite 300
Irvine, CA 92715
(714) 955-2555

Circle no. 220 on reader service card.

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

653         strcpy( envstr[8], buf, MAXLINE );
654     else
655         envstr[8] = '\0' ;
656
657     setenv( envstr , 0 );
658
659     /*      Truncate the command line itself (not the environment
660     *      variable, at 127 characters and then echo it to the
661     *      screen if Echo is set. Then extract the program name
662     *      portion of the string (with the next() call).
663     */
664
665     buf[DOSMAXLINE-3] = '\0';
666
667     if( Echo )
668         puts( buf );
669
670     name = next( $buf, ' ', -1 );
671
672
673     /*      Now try to spawn a new process. Suspend the shell until
674     *      task returns.
675     */
676
677     if( (rval = spawnlp(P_WAIT, name, name, buf, NULL)) < 0 )
678     {
679         /*      If we can't find a .com or .exe file then
680         *      see if we can find a batch file with ole
681         *      right name. Otherwise print an error message
682         *      Note that we have to modify the CMDLINE
683         *      environment string so that the leftmost
684         *      argument (will be argv[0] in the child
685         *      process) is the string "sh"
686         */
687
688         if( errno == ENOENT )
689         {
690             sprintf( envstr, "CMDLINE=sh %s %s", name, buf );
691             rval = spawnlp(P_WAIT, "sh", "sh", name, buf, NULL);
692         }
693         if( rval < 0 )
694         {
695             printf("sh: Can't execute %s %s", name, buf);
696             perror(" ");
697         }
698     }
699
700     END_TRACE("execute");
701     return rval;
702 }
703
704 /*-----*/
705
706 int      exp_vars( dest, src, maxcount, mode )
707 char *dest, *src;
708 {
709     /*      Copy src into dest, expanding shell variables as
710     *      appropriate. Shell variables all have a $ or a % as their
711     *      first character. The global pointers Numv and Numc keep
712     *      track of arguments for $<num> variables.
713     *
714     *      If mode == 1    aliases are expanded
715     *      If mode == 2    %args are expanded
716     *      If mode == 3    both are expanded.
717     *
718     *      return the size of the expanded string.
719     *      A mode 1 or 3 call will return with the high bit of src
720     *      set.
721     */
722
723     register int    num;
724     register char *p ;
725     char            *start_dest = dest;
726
727     TRACE("exp_vars");
728     DIAG("exp_vars, input = <%s>\n", src );
729
730
731     /*      Expand aliases. First, remember the original start of
732     *      the target array. Then, set the high bit of *src so that
733     *      getvar will look for an alias. Then actually expand it.
734     *      Then clear the high bit of the first character of dest
735     *      (which will be set if no alias was found).
736     *      There is a second-order recursion here in that getvar()
737     *      makes a mode 2 exp_vars call.
738     */
739
740     if( mode & 1 )
741     {
742         *src |= 0x80;
743         getvar( &src, &dest, &maxcount );
744         DIAG("exp_vars, expanded aliases <%s>\n", start_dest);
745     }
746
747     /*      Now, expand shell variables. If we see an escaped
748     *      character copy both the \ and the character to dest.
749     *      else if the character isn't a shell variable (doesn't
750     *      have a leading $ or %) just copy it. Else, try to
751     *      expand the shell variable.
752     */
753
754     while( *src && maxcount > 0 )
755     {
756         if( *src == '\\' && src[1] )
757         {
758             /* If the character following the \ is a

```



```

760      * $ of % then strip the backslash and copy
761      * the $ or % to the dest array. Otherwise
762      * copy both the \ and the character that
763      * follows.
764      */
765
766      src++;          /* Skip past the \ */
767
768      if( !ISVAR(*src) )
769      {
770          if( --maxcount <= 0 )
771              break;
772          *dest++ = '\\';
773      }
774
775      if( --maxcount <= 0 )
776          break;
777      *dest++ = *src++;
778  }
779  else if( !ISVAR( *src ) || !(mode & 2) )
780  {
781      /* Either character is just a normal character
782      * (not a shell variable) or we're told to not
783      * expand shell variables.
784      */
785
786      *dest++ = *src++;
787      if( --maxcount <= 0 )
788          break;
789  }
790  else if( digit( *++src ) )
791  {
792      /* Expand a $<num> arg. first extract the
793      * <num> from source, then copy the correct
794      * vector out of the Numv array.
795      * If the Numv entry is NULL, don't
796      * put anything in the dest array.
797      */
798
799      for( num = 0; digit(*src); )
800          num = (num * 10) + (*src++ - '0');
801
802      if( num < Numc )
803          for( p = Numv[num]; *p; *dest++ = *p++ )
804              if( --maxcount <= 0 )
805                  break;
806  }
807  else
808  {
809      /* We've found a $ not followed by a number */
810
811      switch( *src++ )
812      {
813          case '*':
814              num = unargv(Numc,Numv,dest,maxcount,' ');
815              dest += num;
816              maxcount -= num;
817              break;
818
819          case 'p':
820              getcwd( dest, maxcount );
821              for( ; *dest ; ++dest, --maxcount )
822                  if( *dest == '\\')
823                      *dest = '/';
824              break;
825
826          case '!': num = get_hnum(); goto skip;
827          case 's': num = Shlev;
828          skip:    if( maxcount > 5 )
829                  {
830                      sprintf( dest, "%d", num );
831                      for( ; *dest ; ++dest, --maxcount )
832                          ;
833                  }
834              break;
835
836          default:
837              --src;
838              getvar( &src, &dest, &maxcount );
839              break;
840      }
841  }
842
843      *dest = '\\0';
844  }
845
846      *start_dest &= 0x7f;
847
848      DIAG("exp vars: on return <%s>," , start_dest );
849      DIAG(" returning %d\n",          dest - start_dest );
850
851      END_TRACE("exp_vars");
852
853      return( dest - start_dest );
854  }
855
856  /*-----*/
857
858  prompt()
859  {
860      /* Print a prompt using the PROMPT environment variable
861      * Return true.
862      */
863
864      char      buf[50];
865      register char *p;
866
867      if( Mode == INTERACTIVE || Echo )
868      {
869          if( !(p = getenv("PROMPT")) )
870              printf( "[%d:%d] ", Shlev, get_hnum() );

```

(Continued on next page)


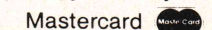
PRIME FEATURES

- Execute DOS level commands in HS/FORTH, or execute DOS and BIOS functions directly.
- Execute other programs under HS/FORTH supervision. (editors debuggers file managers etc)
- Use our editor or your own.
- Save environment any time as .COM or .EXE file.
- Eliminate headers, reclaim space without recompiling.
- Trace and decompile.
- Deferred definition, execution vectors, case, interrupt handlers.

HS/ FORTH

- Full 8087 high level support. Full range transcendentals (tan sin cos arctan logs exponentials)
- Data type conversion and I/O parse/format to 18 digits plus exponent.
- Complete Assembler for 8088, 80186, and 8087.
- String functions - (LEFT RIGHT MID LOC COMP XCHG JOIN)
- Graphics & Music
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management
- Full megabyte - programs or data
- Fully Optimized & Tested for: IBM-PC XT AT and JR COMPAQ and TANDY 1000 & 2000 (Runs on all true MSDOS compatibles!)
- Compare
BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
with AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 55-140 sec
FASTEST FORTH SYSTEM AVAILABLE.
TWICE AS FAST AS OTHER FULL MEGABYTE FORTHS!
(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$270.

 Visa  Mastercard

HARVARD SOFTWARES

P.O. BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Circle no. 132 on reader service card.

Instant-C:TM The Fastest Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only **Instant-C** saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with **Instant-C**. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. **Only Instant-C will let you edit, test, and debug at the fastest possible speeds.**

Fastest Testing. **Instant-C** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. **Instant-C** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

Fastest Programming. **Instant-C** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86. **Instant-C gives you working, well-tested programs faster than any other programming tool.** Satisfaction guaranteed, or your money back in first 31 days. **Instant-C** is \$495.

Rational
Systems, Inc.

P.O. Box 480
Natick, MA 01760
(617) 653-6194

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

871         else
872         {
873             exp_vars( buf, p, 50, 2 );
874             printf( buf );
875         }
876     }
877     return 1;
878 }
879 }
880
881 /*-----*/
882
883 int     docmd( cmd )
884 char    *cmd ;
885 {
886     /*      Do one command from line.
887      *      Return an exit status (or -1 on error).
888      */
889
890     register DIRECTORY *dp;
891     register int      rval = -1;
892
893     TRACE("docmd");
894     PSTR( "docmd(1) cmd=", cmd );
895
896     /* If there're no wild cards in cmd then just strip backslashes
897      * and quotes. Else, expand the wild cards ( exp_dir will strip
898      * backslashes etc.) Then execute the command.
899      */
900
901     if( !has_wild(cmd) )
902         strip(cmd);          /* Just strip backslashes */
903
904     else if( !exp_dir(cmd, MAXLINE) )
905         goto abort;
906
907     rval = execute(cmd) ;
908
909 abort:  END_TRACE("docmd");
910     return( rval );
911 }
912
913 /*-----*/
914
915 rcopy( dest, src, maxcount )
916 register char *dest ;
917 char          *src ;
918 {
919     register char *tbuf;
920     char          *tail;
921     char          *sd = dest;
922
923     DIAG("rcopy (top of proc): src = < %s>\n", src );
924
925     if( !*src || maxcount <= 0 )
926         return;
927
928     if( !(tbuf = malloc( maxcount )) )
929         fprintf(stderr, "Sh: out of memory\n");
930     else
931     {
932         /* 1) expand the src buffer into tbuf
933          * 2) advance tail to point past the next ; and replace
934          *    the ; with a null.
935          * 3) copy everything up to the tail to dest and add a
936          *    ; to dest if the tail is non-null.
937          * 4) repeat this process using the tail as source.
938          */
939
940         /*1*/ exp_vars( tbuf, src, maxcount, 3 );
941
942         tail = tbuf;
943         next( &tail, ' ', '\\');
944         SKIPWHITE( tail );
945
946         /*3*/ for(src = tbuf; *src && src-tbuf < maxcount; *dest++ = *src++)
947             ;
948
949         if( (maxcount -- (src-tbuf)) && *tail )
950             *dest++ = ' ';
951
952         *dest = '\0';
953
954         DIAG("rcopy (before recursive call): dest = < %s>\n", sd );
955
956         /*4*/ rcopy(dest, tail, maxcount);
957         free(tbuf);
958
959         DIAG("rcopy (after recursive call): dest = < %s>\n", sd );
960     }
961 }
962
963 /*-----*/
964
965 char    *next_cmd()
966 {
967     /*      Get a line from input, split off one command, and
968     *      then return a pointer to it (or NULL if at end of
969     *      input. History processing is done here too.
970     *      Semicolons inside quoted strings or preceded by a \
971     *      do not separate commands. Comments and blank lines
972     *      are absorbed (ie. next_cmd won't return until it
973     *      gets a real input line).
974     */
975
976     static char  Cmdbuf[MAXLINE]; /* Should initialize to zeros */

```



```

977 static char *src = Cmdbuf ;
978 register char *p ;
979 char *tbuf;
980
981 TRACE("next_cmd");
982
983 DIAG("next_cmd: src is <%s>\n", src );
984
985 while( !*src )
986 {
987     do
988     {
989         prompt();
990         if( !(*Ifunct)() )
991         {
992             END_TRACE("next_cmd" );
993             return NULL;
994         }
995
996         DIAG("next_cmd: got <%s> from input\n", Ibuf );
997
998         src = Ibuf;
999         SKIPWHITE( src );
1000     }
1001     while( !*src || *src == COMMENT );
1002
1003     history( src, MAXLINE - (src-Ibuf) );
1004     if( *src )
1005     {
1006         rcopy( Ibuf, src, MAXLINE );
1007         src = Ibuf;
1008     }
1009 }
1010
1011 p = next( &src, ' ', '\\ ' );
1012
1013 DIAG("next_cmd: buffer <%s>\n", p );
1014 DIAG("next_cmd: on next call buffer will be <%s>\n", src );
1015
1016 strcpy( Cmdbuf, p );
1017 if( Verbose )
1018     printf("[sh %d input] <%s>\n", Shlev, Cmdbuf );
1019
1020 DIAG("next_cmd: returning <%s>\n", Cmdbuf );
1021 END_TRACE("next_cmd" );
1022
1023 return( Cmdbuf );
1024 }
1025 }
1026
1027 /*-----*/
1028 char *errmsgs[] =
1029 {
1030     "sh
1031     sh %ci
1032     sh %cc <string>
1033     sh file args...
1034     sh %cq
1035     sh %cv
1036     sh %cx
1037     ""
1038 };
1039
1040 /*
1041     shell entered in interactive mode\n",
1042     enter interactive mode\n",
1043     commands are read from string. If several strings\n",
1044     are present they are concatenated together before\n",
1045     execution. The 'c' may be upper or lower case.\n",
1046     commands are taken from <file>. Args are expanded\n",
1047     to correspond with $0 $1 etc inside the file.\n",
1048     for DOS compatability $0 $1 etc are also recognized\n",
1049     Strip quotes from quoted argument strings. Usually\n",
1050     they are left in so that a spawned process can\n",
1051     assemble its argv correctly.\n",
1052     Print input lines to the shell as they are read.\n",
1053     Print lines as they are executed\n",
1054 */
1055
1056 usage( c )
1057 {
1058     /*
1059     Print the usage error message.
1060     */
1061
1062     register char **pp;
1063
1064     fprintf(stderr, "Sh: illegal argument <%c>: Usage is\n\n", c);
1065     for( pp = errmsgs; *pp; fprintf(stderr, *pp++, Switchar) )
1066     ;
1067     exit( 1 );
1068 }
1069
1070 /*-----*/
1071 int setargs( p )
1072 register char *p;
1073 {
1074     /*
1075     Set various global flags base on command line
1076     arguments. This routine will also be used by
1077     "set" which can't recognize the -c switch. So,
1078     ignore -c if c.enabled is false. p should point
1079     at the - on entry. Processing will terminate when
1080     a space or tab or end of string is found.
1081     */
1082
1083     TRACE("setargs");
1084
1085     if( **p )
1086     {
1087         for( ; *p; p++ )
1088         {
1089             switch( *p )
1090             {
1091                 case 'c':
1092                 case 'C': Mode = COMMAND; break;
1093                 case 'q': Noquotes = 1; break;
1094                 case 'v': Verbose = 1; break;
1095                 case 'x': Echo = 1; break;
1096             }
1097         }
1098     }
1099 }

```

(Continued on next page)

LISP

FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

DATA TYPES

Lists and Symbols
Unlimited Precision Integers
Floating Point Numbers
Character Strings
Multidimensional Arrays
Files
Machine Language Code

MEMORY MANAGEMENT

Full Memory Space Supported
Dynamic Allocation
Compacting Garbage Collector

FUNCTION TYPES

EXPR/FEXPR/MACRO
Machine Language Primitives
Over 190 Primitive Functions

IO SUPPORT

Multiple Display Windows
Cursor Control
All Function Keys Supported
Read and Splice Macros
Disk Files

POWERFUL ERROR RECOVERY

8087 SUPPORT

COLOR GRAPHICS

LISP LIBRARY

Structured Programming Macros
Editor and Formatter
Package Support
Debugging Functions
.OBJ File Loader

RUNS UNDER PC-DOS 1.1 or 2.0

IQLISP

5¼" Diskette
and Manual _____ \$175.00

Integral Quality

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

UNIVERSAL CROSS-REFERENCER

• IT WORKS WITH **ALL** LANGUAGES

BASIC, C, PASCAL, FORTRAN, COBOL,
ASSEMBLER, dBASE . . . you name it.

• IT HANDLES standard languages, extended languages & exotic languages.

• IT CONFIGURES to your compiler, interpreter, or assembler — ALL BRANDS — ALL VERSIONS.

INTRODUCTORY PRICE **\$39⁹⁵**

\$3.00 S/H - MC/Visa/Check - 6% Texas Sales Tax
For the IBM PC, XT & compatibles. DOS 2.0+

DALSOFT SYSTEMS

3565 High Vista - Dallas, TX 75234
(214) 247-7695

Circle no. 86 on reader service card.

Quelo® 68000 Software Development Tools

68000/68010 Assembler Package

Assembler, linker, object librarian and extensive indexed
typeset manuals.

Conforms to Motorola structured assembler, publication
M68KMASM[4]. Macros, cross reference and superb load
map, 31 character symbols.

Optimized for CP/M-80, -86, -68K, MS-DOS, PC-DOS \$ 595

Portable Source in "C" \$3000

Lattice® 68000 "C" Cross Compiler

and Quelo 68000/68010 Assembler Package

Optimized for MS-DOS \$1095

68200 Assembler Package

Optimized for CP/M-80, MS-DOS, PC-DOS \$ 595

68020 Assembler Package

Optimized for CP/M-68K, MS-DOS \$ 750

Portable Source in "C" \$3500

For more information contact

Quelo Inc.

2464 33rd W. Suite #173

Seattle, WA 98199

Patrick Adams

Phone (206) 285-2528

COD, Visa, MasterCard

Telex II (TWX) 910-333-8171

CP/M, tm DRI, MS-DOS tm Microsoft, Lattice, tm Lattice Inc.

Circle no. 205 on reader service card.

HISOFT

HIGH QUALITY SOFTWARE CP/M

HiSoft has been selling Z80 CP/M software in
Britain and Europe for over 4 years. Now we'd
like to introduce you to our range of program-
ming languages:

HiSoft Devpac: Z80 assembler/editor/
debugger

HiSoft C: Kernighan/Ritchie
implementation

HiSoft Pascal: fast, standard compiler
All at \$69 inclusive each.

These programs are also available for other
Z80 machines including Timex 2068.

Call or write for full technical details and
press commentaries, or order from:



HISOFT

180 High St. North
Dunstable LU6 1AT
ENGLAND
01144 (582) 696421

Circle no. 134 on reader service card.

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

1087         case 'i': Mode    = INTERACTIVE;      break;
1088         case ' ':
1089         case '\t':
1090         default:  usage( *p );
1091         }
1092     }
1093 }
1094 abort:
1095     END_TRACE("setargs");
1096 }
1097
1098 /*-----*/
1099
1100 doargs(argc, argv)
1101 char **argv;
1102 {
1103     if( --argc <= 0 )
1104     {
1105         Mode = INTERACTIVE;
1106         Ifunct = interactive_input;
1107         return;
1108     }
1109
1110     if( **(++argv) == Switchchar )
1111     {
1112         setargs( *argv++ );
1113         --argc;
1114         /* There's at least one arg,
1115          * skip to second arg and call
1116          * set args if the first char
1117          * is a -. Then skip past the
1118          * second arg too.
1119         */
1120
1121         Numv = argv ;
1122         Numc = argc ;
1123
1124         if( argc <= 0 || !**argv )
1125         {
1126             fprintf(stderr, "Sh: missing file name, ");
1127             fprintf(stderr, "use -i for interactive input\n");
1128             exit(1);
1129         }
1130         else if( Mode == COMMAND )
1131         {
1132             Ifunct = command_input;
1133         }
1134         else if( Mode == FILEMODE && !(Filename = search(*argv, "bat")))
1135         {
1136             fprintf(stderr, "Sh: can't find <S>\n", *argv );
1137             exit(1);
1138         }
1139     }
1140 }
1141
1142 /*-----*/
1143
1144 setenv( env, allocate )
1145 char *env;
1146 {
1147     /* Set an environment. Env may be "name=contents" or the - may
1148     * be a space.
1149     * If allocate is true allocate space, otherwise acutally use
1150     * env (which must be static) as the string.
1151     */
1152
1153     register char *p;
1154
1155     /* Look for either a space or a - in the string. If you
1156     * find a space, replace it with an -.
1157     */
1158
1159     for( p = env ; *p && *p != ' ' && *p != '-' ; p++ )
1160     {
1161         if( *p == ' ' )
1162             *p = '-';
1163     }
1164
1165     /* Now set the environment to the indicated value
1166     */
1167
1168     if( p = allocate ? strsave( env ) : env )
1169         if( putenv(p) != -1 )
1170             return;
1171
1172     fprintf(stderr, "Sh: setenv failed, out of memory\n");
1173     if( allocate && p )
1174         free(p);
1175 }
1176
1177 /*-----*/
1178
1179 set( str )
1180 register char *str;
1181 {
1182     /* Set a shell variable, syntax is
1183     * [whitespace] <name> [= <value>]
1184     */
1185
1186     register int i;
1187     char *name;
1188
1189     /* Get the name, replacing the trailing blank or - with
1190     * a null. Print variables if there is no name.
1191     */
1192
1193     if( !*str )
1194     {
1195         printf("%-8s: %s\n", "echo", Echo ? "ON" : "OFF");
1196         printf("%-8s: %s\n", "cmd", Cmd ? "ON" : "OFF");
1197         printf("%-8s: %s\n", "verbose", Verbose ? "ON" : "OFF");
1198         printvars();
1199     }
1200     else

```



```

1194 {
1195     for(name = str; *str; ++str)
1196         if( *str == '-' || *str == ' ' )
1197             {
1198                 *str++ = 0;
1199                 break;
1200             }
1201
1202     /*      Get the command tail, skipping past any -
1203     *      or blanks to get there.
1204     */
1205
1206     while( *str == '-' || *str == ' ' )
1207         ++str;
1208
1209     /*      Process the command:
1210     */
1211
1212     i = *str ? atoi(str) : 1;
1213
1214     if( !strcmp( "echo", name) ) Echo = i;
1215     else if( !strcmp( "verbose", name) ) Verbose = i;
1216     else if( !strcmp( "cmd", name) ) Cmd = i;
1217     else setvar(name, str);
1218 }
1219 }
1220
1221 /*-----
1222 *      Alias support uses the same tables as shell variables. However,
1223 *      the top bit of the first character of the alias name is set
1224 *      to indicate its function.
1225 */
1226
1227 unalias( str )
1228 char *str;
1229 {
1230     if( *str )
1231     {
1232         *str |= 0x80;
1233         unsetvar( str );
1234     }
1235 }
1236
1237 alias( str )
1238 char *str;
1239 {
1240     if( !*str )
1241         printalias();
1242     else
1243     {
1244         *str |= 0x80;
1245         set( str );
1246     }
1247 }
1248
1249 /*-----*/
1250
1251 void pwd()
1252 {
1253     /*      Print working directory
1254     */
1255
1256     char nbuf[80];
1257
1258     if( !getcwd( nbuf, 80 ) )
1259         fprintf(stderr, "sh: path too long to print.\n");
1260
1261     printf("%s\n", nbuf );
1262 }
1263
1264 /*-----*/
1265
1266 disk_present( id )
1267 int id;
1268 {
1269     /*      Return true if there's a disk plugged into the
1270     *      indicated drive, else print an error message
1271     *      and return 1. This routine assumes that drive
1272     *      C has a disk in it so it will return true if id == 'c'
1273     *      or id == 'C' without checking.
1274     */
1275
1276     register int try = 5; /* times to try to read disk */
1277     register int err;
1278     union REGS regs;
1279
1280     if( (id = toupper(id)) == 'C' || id == 'c' )
1281         return 1;
1282
1283     regs.h.ah = 4; /* Service #4 (verify sec) */
1284     regs.h.al = 1; /* # of sectors */
1285     regs.x.cx = 0; /* track # & sector # */
1286     regs.h.dh = 0; /* head # */
1287     regs.h.dl = id - 'A'; /* drive # */
1288
1289     /*      Actually read the disk. Loop until we've gotten a timeout
1290     *      error (0x80) from dos try times.
1291     */
1292
1293     do{
1294         err = int86(0x13, &regs, &regs) & 0xff;
1295     } while( (err & 0x80) && (--try >= 0) );
1296
1297     if( err )
1298     {
1299         regs.h.ah = 0x0; /* Recalibrate diskette system */
1300         int86(0x13, &regs, &regs);
1301         fprintf(stderr, "Cd: can't log on drive %c, ", toupper(id));
1302     }
1303 }
1304

```

(Continued on next page)

We will do whatever it takes to make DSD86 the best debugger available for the IBM PC.

For starters, we have by far the best design, a superior base to build from.

While the competition adds new "modes" for every feature, we have a pure, consistent and expandable design. While the competition forces you to accept their particular philosophy, we offer maximized flexibility. If you already have a debugger or are looking for your first, look no further because you can't do any better. We invite you to compare our debugger, DSD86, with any other on the market.

- Recursive Command Macros & Files
- Bind Macros to any key
- Multi-segment Symbol Support
- Symbolic Register & Stack Displays
- User Customizable Screen Layout
- Superior Mode-less Design
- Source Window for MS Languages
- User Writable Commands & Displays
- Fast Screen Update
- Unique Breakpointing Facilities
- 30 Day Money Back Guarantee

Call or write for our free report on truly advanced debugging technology which explains DSD86's design and why it is superior to the debugger you are currently using.

Take the DSD challenge: secure a money back guarantee with any of our competitors. Buy both debuggers and use them for a month. Send the one you like least back for a refund.

Only \$69.95!

Soft Advances
P.O. Box 49473
Austin, Texas 78765
512-478-4763

"Programming for Productivity and Profit"
Please include \$4 shipping



Thinking about C?

**Stop Thinking-
Start Programming Today!**

SPECIAL INTRODUCTORY OFFER!
C' Prime, **Personal Computing and C**,
Plus Apprentice C. **\$99**
A \$169 value only

NEW FROM MANX AZTEC!

C' Prime ~~\$99~~ **\$79**

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

C Apprentice ~~\$49.95~~ **\$39.95**

Learn C quickly with this complete, easy-to-use C language interpreter. **Apprentice C** includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

NEW FROM ASHTON-TATE!

Personal Computing and C

A detailed, easy-to-understand guide to C programming prepared especially by **Ashton-Tate** for use with the new **Aztec C' Prime**. Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

TECWARE
1-800-TEC-WARE

(In NJ call 201-530-6307)



C CHEST LISTING

(Listing Continued, text begins on page 18)

```

1305         fprintf(stderr, "DOS error code = 0x%02x\n", err );
1306     }
1307
1308     return( !err );
1309 }
1310
1311 /*-----*/
1312
1313 void      cd( name )
1314 register char *name;
1315 {
1316     /*      Change the current directory to the indicated name.
1317     *      Log in a new disk if necessary. This routine is
1318     *      a bit more sophisticated than DOS itself, in that
1319     *      it checks if a disk exists before trying to
1320     *      log it on. This checking is done using the
1321     *      "Get diskette status" service of BIOS interrupt 0x13.
1322     *      Get to the current directory on another disk by saying
1323     *      "cd x:" Get to another directory on another disk by
1324     *      saying "cd x:/dir/subdir/etc"
1325     */
1326
1327     if( *name && name[1] == ':' )
1328     {
1329         if( !disk_present(*name) )
1330             return;
1331         else
1332         {
1333             bdos( 0xe, toupper(*name) - 'A', 0);
1334             name += 2;
1335         }
1336     }
1337
1338     if( *name && chdir( name ) < 0 )
1339         fprintf(stderr, "sh: Can't find %s\n", name );
1340 }
1341
1342 /*-----*/
1343
1344 shift()
1345 {
1346     /*      Process the "shift" command (move all the $ args left
1347     *      one notch).
1348     */
1349
1350     if( Numc )
1351     {
1352         --Numc;
1353         ++Numv;
1354     }
1355 }
1356
1357 /*-----*/
1358
1359 doenv()
1360 {
1361     /*      Reads and initializes the various environment variables.
1362     *      This routine should only be called once and it must be
1363     *      called before Shlev is used and before command line
1364     *      processing is done.
1365     */
1366
1367     static char      sbuf[16];
1368     register char      *p;
1369
1370     if( (p = getenv("SWTCHAR")) && *p )
1371         Switchar = *p ;
1372
1373     if( (p = getenv("SHLEV")) && *p )
1374         Shlev = atoi(p);
1375
1376     sprintf(sbuf, "SHLEV=%d", ++Shlev );
1377     setenv (sbuf);
1378 }
1379
1380 /*-----*/
1381
1382 use_exit()
1383 {
1384     /*      We get here on a SIGINT (^C) interrupt
1385     */
1386
1387     signal( SIGINT, use_exit );
1388     fprintf(stderr, "Use \"exit\" or \"logout\" to leave outer shell.\n");
1389 }
1390
1391 /*-----*/
1392
1393 TOKEN tokenize( buf )
1394 register char *buf;
1395 {
1396     /*      This is an extremely primitive token recognizer that will
1397     *      eventually be replaced with something more reasonable.
1398     */
1399
1400     if( !strcmp( "alias",      buf) ) return ALIAS;
1401     if( !strcmp( "cd",         buf) ) return CD;
1402     if( !strcmp( "exit",       buf) ) return EXIT;
1403     if( !strcmp( "history",    buf) ) return HISTORY;
1404     if( !strcmp( "logout",     buf) ) return LOGOUT;
1405     if( !strcmp( "pwd",        buf) ) return PWD;
1406     if( !strcmp( "rem",        buf) ) return REM;
1407     if( !strcmp( "setenv",     buf) ) return SETENV;
1408     if( !strcmp( "set",        buf) ) return SET;
1409     if( !strcmp( "shift",      buf) ) return SHIFT;
1410     if( !strcmp( "unalias",    buf) ) return UNALIAS;
1411     if( !strcmp( "unset",      buf) ) return UNSET;

```

(Continued on page 98)

INTRODUCING DATALIGHT C

The Datalight C Compiler for MSDOS is a full C with all K&R constructs, including bitfields, plus the version 7 extensions. Other features of the compiler are:

\$60

- Produces object files (.obj) so just the MSDOS linker is required.
- Floating point performed with 8087 or automatic software floating.
- Over 100 compact library functions with source.
- Compatible with the Lattice C compiler.
- Runs on IBM-PC, and compatibles, running MSDOS 2.0 or later.
- Complete, easy-to-read users' manual with index.
- Highly optimized code for production quality programs.

Datalight

11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

Outside USA add \$10 shipping. Washington State residents add 7.9% sales tax. VISA and MasterCard accepted.

IBM-PC, a trademark of IBM; MSDOS, a trademark of Microsoft Corp.; Lattice C, a trademark of Lattice Corp.

Circle no. 203 on reader service card.

INDUSTRIAL PASCAL FOR THE 68000

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming critical that a high level language be used whenever possible. Find out why over 1400 companies have switched to OmegaSoft Pascal for their demanding applications.

OmegaSoft Pascal takes the Pascal framework and expands the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The compiler package includes the compiler, interactive symbolic debugger, relocatable macro assembler, linking loader, and screen editor. Source code is provided for the debugger, screen editor and runtime library.

Versions to run under the OS-9/68000 and VERSAdos operating systems are currently available to end-users and OEM's. End user price is \$900 (domestic) or \$925 (international). A version for CP/M-68K is available for OEM use, with OEM versions for UNIX type operating systems to follow.

T.M. OmegaSoft is a trademark of Certified Software Corporation. OS-9/68000 is a trademark of Microware. VERSAdos is a trademark of Motorola. CP/M-68K is a trademark of DRI. UNIX is a trademark of Bell Labs.

CERTIFIED SOFTWARE CORPORATION

616 Camino Caballo, Nipomo, CA 93444
Telephone: (805) 929-1395; Telex: 467013

Circle no. 209 on reader service card.

MORE... VALUE and PERFORMANCE with Mitek's Relocatable Z80 Macro Assembler and Z80 Symbolic Debugger

NEW Z80 Symbolic Debugger

- Only \$49.95 plus shipping.
- Screen oriented with a simultaneous display of instruction mnemonics, register, stack, and memory values.
- Breakpoints may be set on any combination of fixed memory address, register values and/or memory values.
- Uses Digital Research compatible SYM files.
- Supports Hitachi HD64180.

Relocatable Z80 Macro Assembler

- Only \$49.95 plus shipping.
- 8080 to Z80 Source Code Converter.
- Generates Microsoft compatible REL files or INTEL compatible hex files.
- Compatible with Digital Research macro assemblers MAC & RMAC.
- Generates Digital Research compatible SYM files.
- Conditional assembly.
- Phase/dephase.
- Cross-reference generation.
- Full Zilog mnemonics.
- INCLUDE and MACLIB FILES.
- Separate data, program, common, and absolute program spaces.
- Supports Hitachi HD64180.
- Z80 Linker and Library Manager for Microsoft compatible REL files available as an add-on to Assembler.

ATTENTION Turbo Pascal Users:

Assembler will generate Turbo Pascal in-line machine code include files.

PRICE LIST

Z80 Macro Assembler: \$49.95
Assembler, Linker, and Library Manager: \$95.00
Manual Only: \$15.00
Z80 Symbolic Debugger: \$49.95
Manual Only: \$15.00
Assembler, Linker, Library Manager, and Debugger: \$134.95

Include \$5 for shipping and handling.

TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: (808) 623-6361

Specify desired 5 1/4" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed.

P. O. Box 2151
Honolulu, HI 96805

MITEK

Z80 is a trademark of Zilog, Inc. MAC and RMAC are trademarks of Digital Research, Inc. Turbo Pascal is a trademark of Borland International, Inc.

Circle no. 190 on reader service card.

A general purpose programming language for string and list processing and all forms of non-numerical computation.

SNOBOL4+ — the entire

SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator

• ASCII, binary, sequential, and random-

access I/O • Assembly Language inter-

face • Compile new code during

program execution • Create

SAVE files • Program

and data space up

to 300K bytes

RAM

With **ELIZA** & over 100 sample programs and functions

SNOBOL4+ • **ARTIFICIAL INTELLIGENCE**

SYMBOLIC MATHEMATICS • **CRYPTOGRAPHY** • **LINGUISTICS** • **LIST PROCESSING** • **GAMES**

For all 8086/88PC/MS-DOS 128K minimum 5 1/4 DSDD.

Send check, VISA, MC for \$95 to:

Prentice-Hall, Inc.

B&P Div. Attn: F. Roess Englewood Cliffs, NJ.
800-624-0023, in New Jersey 800-624-0024

Circle **no. 201** on reader service card.



Users' Group

Over 60 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

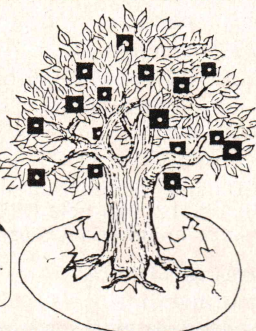
The C Users' Group

Post Office Box 97
McPherson, KS 67460
(316) 241-1065

Circle **no. 181** on reader service card.

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



COGITATE

"A Higher Form of Software"

24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345

TELEX: 386581 COGITATE USA

Circle **no. 81** on reader service card.

C CHEST LISTING

(Listing Continued, text begins on page 18)

```

1412         return CMD;
1413     }
1414     /*-----*/
1415     cmds()
1416     {
1417         /*
1418          * Process commands using the current ifunct. Continue till
1419          * end of input is reached. If expand_vars is false then
1420          * aliases and $args aren't expanded. This lets us read
1421          * in the shrc.bat file unmolested
1422          */
1423
1424         char          cmdbuf[9];          /* Place to put extracted command */
1425         char          *p, *start;
1426         register char *cmd;
1427         register int i;
1428         int           rval = 0;
1429
1430         while( start = cmd = next_cmd() )
1431         {
1432             DIAG("cmds: next_cmd returned <%=>\n", cmd );
1433
1434             /*
1435              * Strip the command name from the rest of the command
1436              */
1437
1438             p = cmdbuf;
1439             for( i = 0; --i >= 0 && *cmd && *cmd != ' ' ; *p++ = *cmd++)
1440             ;
1441             *p = '\0';
1442
1443             DIAG("cmds: partitioned <%=>", cmdbuf );
1444             DIAG("%+ <%=>\n", cmd );
1445
1446             SKIPWHITE(cmd);
1447
1448             /* If Echo is set then echo the command to standard
1449              * output. Note that if i == CMD then the command
1450              * will be echoed in execute() (called by docmd()).
1451              */
1452
1453             if( (i = (int) tokenize(cmdbuf)) != (int) CMD && Echo )
1454                 puts( start );
1455
1456             switch( i )
1457             {
1458             case ALIAS:      alias ( cmd );          break;
1459             case CD:         cd ( cmd );             break;
1460             case CMD:        rval = docmd(start);     break;
1461             case HISTORY:    print_hist( stdout );    break;
1462             case PWD:        pwd ( );                break;
1463             case REM:        break;                  break;
1464             case SET:        set ( cmd );             break;
1465             case SETENV:    setenv ( cmd );          break;
1466             case SHIFT:      shift ( );              break;
1467             case UNALIAS:    unalias ( cmd );        break;
1468             case UNSET:      unsetvar( cmd );        break;
1469             default:         printf ("Illegal token\n"); break;
1470
1471             case LOGOUT:
1472                 reset_fileinput();
1473                 if( access(Filename = "/logout.bat", 04) == 0 )
1474                     cmds();
1475
1476                 /* Fall through to EXIT case */
1477
1478             case EXIT:
1479                 goto exit;
1480
1481             }
1482         }
1483     exit:
1484         return rval;
1485     }
1486     /*-----*/
1487     main(argc, argv)
1488     char **argv;
1489     {
1490         /*
1491          * Exit status is that of the most
1492          * recently excuted program. In the case of a batch file
1493          * the exit status will be that of the shell processing
1494          * the batch file (ie. of the last program executed by
1495          * that shell).
1496          */
1497
1498         reargv(&argc, &argv); /* Get a long command line if one's there*/
1499         doenv();               /* Read various environment variables and
1500                                * initialize Shlev, which holds the
1501                                * current shell level.
1502                                */
1503
1504         /*
1505          * Process the two automatic batch file. /shrc.bat is
1506          * executed every time a shell is created. /login.bat
1507          * is only executed when a level 0 shell is created.
1508          */
1509
1510         if( access(Filename = "/shrc.bat", 04) == 0 )
1511         {
1512             cmds();
1513             reset_fileinput();
1514         }
1515
1516         if( Shlev == 0 && access(Filename = "/login.bat", 04) == 0 )
1517         {
1518             cmds();
1519         }
1520     }

```



```

1519         reset_fileinput();
1520     }
1521
1522
1523     /*      Now process the command line arguments. Doargs will
1524     *      set the Echo, Verbose, Cmd and Mode variables as
1525     *      appropriate. If we're in an interactive, level
1526     *      0 shell, Call signal to prevent ^C from working on
1527     *      the shell itself and print a copyright notice.
1528     */
1529
1530     doargs(argc, argv);          /* Process command line args */
1531
1532     if( Shlev == 0 && Mode == INTERACTIVE )
1533     {
1534         signal( SIGINT, use_exit );
1535         fprintf(stderr,"SH (ver %s) - Copyright (c) 1985, ", VER);
1536         fprintf(stderr,"Allen I Holub. All rights reserved.\n");
1537     }
1538
1539
1540     /*      Finally, process commands from the input source determined
1541     *      by the command line.
1542     */
1543
1544     exit( cmds() );
1545 }

```

End Listing

Shell listings continue next month

Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into **great** computers.

SEMIDISK

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075 503-642-3100

Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards (300/1200 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Battery Backup, Too

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

The Best News

	<u>512K</u>	<u>1Mbyte</u>	<u>2Mbyte</u>
SemiDisk I, S-100	\$695	\$1395	
SemiDisk II, S-100	\$995		\$1995
IBM PC, XT, AT	\$595		\$1795
QX-10	\$595		\$1795
TRS-80 II, 12, 16	\$695		\$1795
Battery Backup Unit	\$150	\$150	\$150

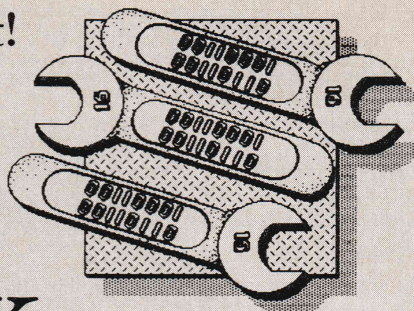
Someday you'll get a SemiDisk.
Until then, you'll just have to....wait.



Z80 CP/M Users Take Heart!
Here's All You Need to Write
Your Own Programs For Only \$25!

DR. DOBB'S TOOLBOOK FOR Z 80

Item #022



Do you use CP/M? Do you feel as if the only part of the computer industry that has *not* abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write them yourself. When you do, you quickly find that while Pascal or Basic is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

DR. DOBB'S TOOLBOOK FOR Z80 puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

- **A method of designing programs and coding them in assembly language**—and—a demonstration of the method in the construction of several complete, useful programs.

- **A complete integrated toolkit of subroutines** for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's run-time library to your assembly language work, without a compiler's size and sluggish code.

Best of all, every line of the toolkit's source code is there for you to read, and every module's operation is explained with the clarity and good humor for which Dave Cortesi's writing is known.

Save Yourself the Time and Frustration of File Entry
Order the Z80 Software on Disk!

All the software in **DR. DOBB'S TOOLBOOK FOR Z80** the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and a Digital RMAC assembler or equivalent are required.)

Receive **DR. DOBB'S TOOLBOOK FOR Z-80**, along with the software on disk together for only \$40. Item #022A

To order, return this form with your payment, plus \$1.75 for shipping in the U.S., \$3.75 outside the U.S., to: M&T Publishing 2464 Embarcadero Way, Palo Alto, CA 94303

Or, for faster service on credit card orders, CALL TOLL FREE 1-800-528-6050, ext. 4001. Refer to item #022 for the **Z80 Toolbook** and item #022A for **The Z80 Toolbook** with the disks. Please specify disk format.

YES! Please send me _____ copies of **DR. DOBB'S TOOLBOOK FOR Z 80** for \$25 each.

Please send me _____ sets of **DR. DOBB'S TOOLBOOK FOR Z-80** together with the software on disk for \$40 per set.

Subtotal _____

CA residents add applicable sales tax _____%

Shipping _____

TOTAL _____

For the Z-80 software on disk, please specify one of the following formats: _____ 8" SS/SD _____ Apple _____ Osborne _____ Kaypro

_____ Check enclosed Charge my _____ VISA _____ M/C _____ Amer. Exp.

Card # _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

PL/68K

LISTING ONE (Text begins on page 26)

```

/*
    Declare the format of a macro table.
*/

struct node {
    struct node * next; /* Pointer to next node. */
    int    nargs;      /* Number of args in macro. */
    char * name;       /* Pointer to name of macro. */
    char * text;       /* Pointer to replacement text of macro. */
};

/*
    Define the hash table used to access nodes of the macro table.
*/

#define MAC_PRIME 101

struct node * ht [MAC_PRIME];

/*
    Look up a symbol in the macro table.

    Output:  2      if found, NZ if not found.
            a0      pointer to text of symbol.
            d0w     number of arguments (0-n), -1 if no arguments.
*/

#define hash_val d2
#define bp0 a1
#define bpl a2

lookup(symbol)
register char * symbol;
{
    register struct node **bp0, *bpl;
    register int hash_val;

    /* Get the hash value of the symbol into hash_val. */
    hash(symbol);
    hash_val = d0;

    /* Point bp0 into the hash table. */
    bp0 = &ht;
    hash_val *= sizeof(struct node *);
    adda(hash_val, bp0);

    /* Search down the list of buckets hanging from the hash table. */
    for (bpl = *bp0; bpl; bpl = bpl->next) {
        str_eq(symbol, bpl->name);
        if (Z) {
            /* Match. */
            a0 = bpl->text;
            d0 = bpl->nargs;
            move(Z_BIT, ccr);
            return;
        }
    }

    /* No match. */
    move(NZ_BIT, ccr);
}

```

End Listing One

LISTING TWO

```

ht:    ds.l        101

lookup:
    movem.l        a1/a2/d2, -(sp) ;function entry
    subq          #4, sp

    move.l         20(sp), (sp) ;get the hash value into hash_val
    jsr           hash
    move.l         d0, d2
    move.l         #ht, a1 ;point bp0 into the hash table
    mulu          #4, d2
    adda.l         d2, a1

;    search down the list of buckets

    move.l         (a1), a2 ;for (bpl = * bp0; ...; ...)
    bra           _3

_1:    move.l         6(a2), (sp) ;str_eq(symbol, bpl -> name);
    move.l         20(sp), -(sp)
    jsr           str_eq
    addq          #4, sp
    bnz          _2
    move.l         6(a2), a0 ;      ;if (Z)
    move.w         4(a2), d0 ;      a0 = bpl -> text;
    move          #4, ccr ;      d0 = bpl -> nargs;
    bra           _4          ;      move(Z_BIT, ccr);
                           ;      return;

_2:    move.l         (a2), a2 ;for (...; ...; bpl = bpl -> next)

_3:    cmpa.l        #0, a2 ;for (...; bp; ...)
    bnz          _1
    move          #0, ccr ;move(NZ_BIT, ccr);

_4:    addq          #4, sp ;function exit
    movem.l        (sp)+, a1/a2/d2
    rts

```

End Listings

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA NO SALES TAX

V20	CPU JPD70108D-8	\$16.00
8087-2	Math Coprocessors	140.00
DYNAMIC RAM		
256K	64Kx4 150 ns	\$4.75
256K	256Kx1 120 ns	3.25
256K	256Kx1 150 ns	2.47
128K	128Kx1 150 ns	3.50
64K	16Kx4 150 ns	2.75
64K	64Kx1 150 ns	1.00
EPROM		
27C256	32Kx8 250 ns	\$7.50
27256	32Kx8 250 ns	4.75
27C64	8Kx8 200 ns	3.75
2764	8Kx8 250 ns	2.50
STATIC RAM		
6264LP-15	8Kx8 150 ns	\$2.99

OPEN 7 DAYS WE CAN SHIP VIA FED-EX ON SAT

NO EXTRA COST FOR F-EX SAT DELIVERY ON ORDERS RECEIVED BY 11:59 AM PT. P-ONE



MasterCard/VISA or UPS CASH CDD
Factory New, Prime Parts 
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421 **(918) 267-4961**

Prices shown above are for Dec. 2, 1985

Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$6.00, or Priority One @ \$13.00!

Circle no. 105 on reader service card.

TOTAL RECALL

FOR \$104.95  

Now you can retrieve everything you entered during program development -- from version one thru all updates. SRMS does it with a complete source utility at an affordable price.

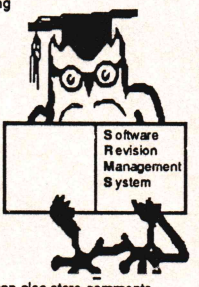
You can retrieve specific versions of a program, make changes and restate the source while recording when, why and where changes were made.

All versions are stored in a single library without duplication of common code or text -- saves disk space. You can also store comments specific to each version -- provides complete development history and documentation of the program.

SRMS supports recall and edit of programs written in BASIC, FORTRAN, PASCAL (including Turbo Pascal), C and ASSEMBLY CODE.

Requires MS or PC DOS, Version 2 and up with 128K and one disk (floppy or hard).

Quilt Computing 7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650



Circle no. 107 on reader service card.

Finally... BSW-Make

Now, an **affordable** and **compatible** implementation of the Unix "make" facility for MS-DOS and VAX/VMS. Eliminate bugs due to forgotten compiles!



BSW-Make rebuilds your software quickly, correctly, and **automatically** after your editing session.

MS-DOS: \$89.95

Every wasted compile costs you time. BSW-Make will save you time, and time is money.

VAX/VMS: \$495.95

Why pay more? BSW-Make costs a fraction of DEC's MMS. BSW-Make will save you time and money!

Prices include postage within the United States. Massachusetts residents add 5% sales tax. MasterCard and Visa accepted.  

The Boston Software Works

120 Fulton Street • Boston, MA • 02109 • (617) 367-6846

MS-DOS is a trademark of Microsoft Corp. VAX and VMS are trademarks of Digital Equipment Corp. Unix is a trademark of AT&T Bell Laboratories.

Circle no. 126 on reader service card.

MULTITASKING OS LISTING

(Text begins on page 44)

```

;
;
;      Terra Nova Communications multi-tasking kernel
;      Initialization and task-switcher
;
;      Note: this is not intended to be a complete listing.  It's only
;      a sample of some of the techniques used in our system.
;

PSECT      Kernel

;
;      External symbols (defined in other code segments)
EXTERN      VecTable,          ;vector table for hardware vector list
            JMPTable,          ;jump table for system calls
            JMPTabLen,         ;length of jump table in longwords
            KernEnd,           ;end of kernel code item in heap
            IOInit,            ;our private I/O initialization routine
            HeapInit,          ;our private heap initialization
            SysInit,           ;system variable initializer
            SysConMon,         ;entry point for system console
                                ;monitor task
            HeapMunger,        ;entry point for heap munger task
            DiskMunger         ;entry point for disk munger task

;
;      Entry points in this module (referenced from elsewhere)
ENTRY      Start,              ;primary entry point to boot our OS
            ConSwitch,         ;main context switcher
            ConSwSleep         ;alternate context switcher (puts
                                ;calling task to sleep)

;
;      Include files (mostly equates)
INCLUDE     SysEqu              ;contains the low-memory absolute
                                ;address equates (jump table, etc)
INCLUDE     HeapDef             ;defines the heap data structure
INCLUDE     SysIO               ;contains hardware I/O equates

;
;      Miscellaneous storage
CodeHeap    DS.L      8          ;heap header for kernel heap item
StackEnd    DS.L      40         ;system stack before tasking starts
StackBegin  DS.L      0          ;top of startup stack area

;
;      Pre-tasking initialization
;      this code works in single-task mode
;      prior to the invocation of the context switcher
Start        ;Initial entry.  Calling operating system is still
            ;alive and kicking at this point.
TakeOver     LEA        ReEntry,A1      ;point to re-entry instruction
            MOVE.L     A1,$20.W         ;move short absolute to the vector
            ;for privilege exceptions
            MOVE       USP,A0           ;try a privileged instruction.  If it
            ;works, then we're in priv. mode.  If not, then trap to
            ;ReEntry and be in privileged mode anyway
ReEntry      LEA        StackBegin,A7   ;set up initial stack

;
;      Turn off all interrupts in the system
;      Note: this is device-specific code.
;      The labels in the operand fields are from our own
;      SysIO include file.
CLR.B       FDCIntMask           ;clear floppy disk & system console
CLR.B       HDIntMask            ;clear hard disk completion int. mask
CLR.B       SerIO1IntMask        ;clear serial boards
CLR.B       SerIO2IntMask

;
;      Initialize the vector table
;      Copy the vectors from an assembled table (in another module)
;      into the actual hardware vector list in low RAM
VecMove      LEA        VecTable,A0     ;source (in another code segment)
            LEA        $0.W,A1         ;destination (begins at $00 0000)
            MOVE       #191,D7         ;192 longwords to move
            MOVE.L     (A0)+,(A1)+     ;move a longword
            DBRA       D7,VecMove      ;repeat till done (fast loop on 68010)

;
;      Copy system routine JMP table from assembled object code (in
;      another module) to low memory jump table, where everyone
;      can get at them.
JPTMove      LEA        JMPTable,A0     ;source
            LEA        System.W,A1     ;dest. (name of first system call in
            ;the jump table.  "System" is from the SysEqu include
            ;file.  It's the context switcher)
            MOVE       #JMPTabLen/4,D7 ;number of longwords to move
            MOVE.L     (A0)+,(A1)+     ;move a longword
            DBRA       D7,JPTMove      ;repeat till done (fast loop on 68010)

;
;      Clear low memory to zero (between jmp table and kernel)
LowClr       LEA        StackEnd,A1     ;point to top of destination
            ;and bottom of destination (end of the jump table)
            LEA        System+JMPTabLen.W,A0
            SUBA       A0,A1           ;calculate the length
            MOVE.L     A1,D7           ;move to D7 for counting
            LSR.L      #4,D7           ;divide by 16 for 16-byte blocks
            CLR.L      (A0)+           ;clear 16 bytes, quickly
            CLR.L      (A0)+
            CLR.L      (A0)+
            CLR.L      (A0)+
            DBRA       D7,LowClr       ;do it until done.

```



```

;      Clear high memory to zero (between kernel and end of RAM)
;      (RAMEnd is first byte beyond RAM, defined in SysEqu)
;      LEA      RAMEnd,A1      ;point to top of destination
;                               ;and bottom of destination (end of the jump table)
;      LEA      KernEnd,A0
;      SUBA     A0,A1           ;calc the length
;      MOVE.L   A1,D7           ;move to D7 for counting
;      LSR.L    #4,D7           ;divide by 16 for 16-byte blocks
HiClr   CLR.L    (A0)+           ;clear 16 bytes, quickly
;      CLR.L    (A0)+
;      CLR.L    (A0)+
;      CLR.L    (A0)+
;      DBRA     D7,HiClr        ;do it until done.

;      Initialize all of the primary I/O devices
;      Note: this is a device specific routine not treated in the article.
;      JSR      IOInit

;      Initialize the heap
;      Note: this is a routine in the heap manager, which creates
;      valid heap headers for the three initial heap items discussed
;      in the text: the deletion below the kernel, the kernel code
;      item, and the deletion above the kernel.
;      JSR      HeapInit

;      Initialize the system zone of low memory
;      Note: this sets up the TCB and master handle arrays, as
;      discussed in the text, as well as initializing the time of day and
;      the date and the other miscellaneous system values.
;      JSR      SysInit

;      Spawn off the initial tasks
;      This will create TCBs and TData items for the tasks, but won't
;      invoke them. They're invoked only by the context switcher.
;      LEA      SysConMon,A0     ;point to system console entry point
;      MOVE.L   #4096,D0         ;tell it how much RAM for TData
;      JSR      Spawn            ;jump through jump table entry
;                               ;("Spawn" is a jump table equate in SysEqu)
;      LEA      HeapMunger,A0    ;spawn the heap munger
;      MOVE.L   #512,D0          ;heap munger's TData size
;      JSR      Spawn
;      LEA      DiskMunger,A0    ;Spawn the disk munger
;      MOVE.L   #8192,D0         ;(TData includes one disk buffer)
;      JSR      Spawn
;
;      LEA      TCB1.W,A2        ;get address of first TCB in array
;                               ;(TCB1 is defined in SysEqu)
;      BRA.S    ConSw1           ;now start the context switcher!

;      Context Switcher: primary version
;      Simple task-switch, nothing fancy.
;      SysFlags is a low-RAM system flag byte, defined in SysEqu.
;      The data structure for the TData item is defined in SysEqu.
;      The data structure for the TCB is defined in SysEqu.
ConSwitch BTST     #StopSys,SysFlags.W ;task switching inhibited?
;      BNE.S    ConSwX           ;yes, exit back to caller
;      MOVE.L   OurTCB(A5),A0    ;get TCB address from TData
;      SUBA.L   A5,SP            ;subtract TData base addr from stack
;      MOVE.L   SP,TCBSP(A0)     ;save relative displacement in TCB

;      MOVE.L   TCBNxt(A0),A2    ;get address of next TCB
ConSw1   MOVE.L   TCBA5(A2),A5    ;get new TData base address
;      MOVE.L   TCBSP(A2),SP     ;get stack relative displacement
;      ADDA.L   A5,SP            ;restore absolute address
ConSwX   RTS                    ;return to next task

;      Context Switcher: alternate version
;      Put the calling task to sleep.
ConSwSleep BTST     #StopSys,SysFlags.W ;task switching inhibited?
;      BNE.S    ConSwX           ;yes, exit back to caller
;                               ;without going to sleep
;      MOVE.L   OurTCB(A5),A0    ;get TCB address from TData
;      SUBA.L   A5,SP            ;subtract TData base addr from stack
;      MOVE.L   SP,TCBSP(A0)     ;save relative displacement in TCB

;      MOVE.L   TCBNxt(A0),A2    ;get address of next TCB
;      MOVE.L   TCBPrev(A0),A1   ;get addr of previous TCB
;      MOVE.L   A2,TCBNxt(A1)    ;close the pointers around the now-
;      MOVE.L   A1,TCBPrev(A2)   ;sleeping task.
;      MOVE.B   #Sleep,TCBState(A0) ;mark it as asleep

;      MOVE.L   TCBA5(A2),A5     ;get new TData base address
;      MOVE.L   TCBSP(A2),SP     ;get stack relative displacement
;      ADDA.L   A5,SP            ;restore absolute address
;      RTS                    ;return to next task

```

End Listing

8080 SIMULATOR

LISTING ONE (Text begins on page 76)

```

*****
*
*      8080 Simulator for MC68000
*
*      With CP/M 2.2 call support, optional tracing and
*      Morrow HDDMA DMA buffer translating.
*
*      Version 1.2 1/21/85 JEC
*      Fixed Extent bug in OPEN logic.
*      Sped up code, sample MAC from 2:13 to 1:40.
*      Now runs at a 1.4 MHz equivalent based on MAC sample.
*
*      Version 1.1 8/29/84 JEC
*      Fixed BDOS call #6 bug.
*
*      Version 1.0 05/25/84 by Jim Cathey
*
*      This program has been written for speed wherever possible,
*      as such tends to be large because of the separate subroutine
*      for each and every opcode of the target processor.
*
*      On an 8MHz 68000 (Compupro) system the simulation speed is
*      a little better than a 1MHz Z-80 when running MAC. The time
*      for a sample assembly was 2:13 for the simulation vs 0:35
*      on a 4MHz Z-80, both systems used identical hard disk systems.
*
*      It is not a complete simulation, as some flag handling
*      isn't quite right, but it is enough to run the programs
*      I wrote it for (DDT, LU, MAC, and Morrow's FORMATHW).
*
*****
text
page
*****
*
*      This file contains the startup routines, the simulator core,
*      tracing code, and the CP/M 2.2 simulation.
*
*****
xdef optabl,flags,mnops
globl mloop,illegl,service
*
*      Conditional assembly flags.
*
trace equ 0 ; Non-zero for trace routine inclusion.
trcdsk equ 0 ; Non-zero for FCB trace routine inclusion.
dmpdsk equ 0 ; Non-zero for register dump in FCB trace.
* !! diskio is in file COM2.S !!
*diskio equ 0 ; Non-zero for special HDDMA support.
*
*
*      Register definitions for the simulation.
*
return equ 016,r ; JMP (return) is fast return to MLOOP.
pseudopc equ 015,r ; 8080's PC is register A5.
opptr equ 014,r ; Pointer to opcode dispatch table.
pseudosp equ 013,r ; 8080's SP is register A3.
flagptr equ 012,r ; Pointer to 8080's flag lookup table is A2.
targbase equ 011,r ; Pointer to 8080's address space is A1.
regs equ 011,r ; Base pointer to 8080's registers is A1.
regcon0e equ 7,r ; Register based constant #0E (for speed).
regcon01 equ 6,r ; Register based constant #01.
regcon0f equ 5,r ; Register based constant #0F.
regconff equ 4,r ; Register based constant #FF.
regf equ 3,r ; 8080's Flags
rega equ 2,r ; 8080's Accumulator
*
*      Note, only leaves D0-D1/A0 for free use by entire
*      program without saving registers for temporary use.
*
bdos .opd 0,$4e42 ; BDOS 'macro'.
bios .opd 0,$4e43 ; BIOS 'macro'.
*
page
*****
*      Initialization and Main Opcode dispatcher.
*
*****
start lea.l target,targbase ; Start of target memory.
ifne trace ; Optional trace code.
bsr entrads ; Enter trace delimiting addresses
* ; if the code is desired.
endc
bsr lodfdos ; Load up the fake FDOS in target mem.
bsr lodregs ; Load the remaining simulation registers
bsr loadcom ; Load the .COM program,
tst d0 ; quit if unsuccessful.

```



```

bsr pspace
bsr pspace
move.b (pseudopc),d1
bsr pbyte
bsr pspace
bsr pspace
moveq #0,d0
move.b (pseudopc),d0
asl.w #2,d0
lea.l mnops,a0
move.l (a0,d0.l),d1
move.l d1,-(sp)
inc.l d1
move #9,d0
bdos
move.l (sp)+,a0
cmp.b #" ",(a0)
beq nooprnd
cmp.b #"C",(a0)
bne notcons
move.b 1(pseudopc),d1
bsr pbyte
bra nooprnd
notcons cmp.b #"A",(a0)
bne nooprnd
move.b 2(pseudopc),d1
bsr pbyte
move.b 1(pseudopc),d1
bsr pbyte
nooprnd bsr pspace
bsr pspace
bsr pspace
movem.l (sp)+,d0-d1/a0
rts

page
*****
*
* Initialization subroutines.
*
*****
lodfdos lea.l fdos,a6 ; Load up the fake FDOS.

```

```

move.l targbase,pseudosp
adda.l #10000,pseudosp
lea.l -256(pseudosp),a0
move.w #fdoslen,d0
lodloop move.b (a6)+,(a0)+
dbra d0,lodloop
lea.l -256(pseudosp),a0
move.l a0,d0
sub.l targbase,d0
move.b #sc3,0(targbase) ; Build BIOS & BDOS jumps.
move.b #sc3,5(targbase)
move.b d0,6(targbase)
rol.w #8,d0
move.b d0,7(targbase)
rol.w #8,d0
add.w #3,d0
move.b d0,1(targbase)
rol.w #8,d0
move.b d0,2(targbase)
move.w #0,-(pseudosp) ; Set up a return stack to exit simulation.
rts

lodregs lea.l optabl,opptr ; Point base register to opcode dispatch
table.

lea.l mloop,return
lea.l flags,flagptr
move.l targbase,pseudopc
adda.l #100,pseudopc ; Start execution at 0100H in target space.
moveq #se,regcon0e ; Set up quick constants.
moveq #s1,regcon01
moveq #sf,regcon0f
move.l #fff,regconff
moveq #0,rega
moveq #0,regf
rts

page
entrads move.l #tracemsg,d1 ; Enter trace address if necessary.
move.w #9,d0
bdos

```

(Continued on next page)

Sidekick for CP/M! Write-Hand-Man

Desk Accessories for CP/M

NEW! Now with automatic screen refresh!

Suspend CP/M applications such as WordStar, dBase, and SuperCalc, with a single keystroke and look up phone numbers, edit a notepad, make appointments, view files and directories, communicate with other computers, and do simple arithmetic. Return to undisturbed application! All made possible by **Write-Hand-Man**. Ready to run after a simple terminal configuration! No installation required.

Don't be put down by 16 bit computer owners. Now any CP/M 2.2 machine can have the power of **Sidekick**.

Bonus! User extendable! Add your own applications.

\$49.95 plus tax (California residents), shipping included! Volume and dealer discounts.

Available on IBM 8 inch and Northstar 5 inch disks. Other 5 inch formats available with a \$5.00 handling charge. CP/M 2.2 required; CP/M 3 not supported.

COD or checks ok, no credit cards

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

Write-Hand-Man trademark of Poor Person Software, CP/M trademark of Digital Research, Sidekick trademark of Borland International, dBase trademark of Ashton-Tate, WordStar trademark of Micropro, SuperCalc a trademark of Sorcim.

Circle **no. 169** on reader service card.

TheMax™

Want unparalleled speed and efficiency from your Macintosh? Get TheMax.

TheMax: a 1.5Mb memory board designed to expand to a full 4Mb of power, *plus* **1.5Mb**

MaxRAM: software to configure your memory two ways: 1Mb of contiguous memory with 400K RAM disk or a 512K Mac with a recoverable 1024K RAM disk, *and*

MaxPrint: print spooler software that lets you work and print at the same time.

TheMax is available now for both the 128K and the 512K Macintosh. Kits and 512K upgrades are also available. Ask your dealer for more information, or contact

MacMemory Inc.

473 MACARA AVE., SUITE 701, SUNNYVALE, CA 94086,
(408) 773-9922.

Macintosh is a trademark licensed to Apple Computer Inc.

Circle **no. 218** on reader service card.

8080 SIMULATOR

LISTING ONE (Listing Continued, text begins on page 76)

```

    bsr atol                ; Get trace start address.
    and.l $ffff,d1
    move.l d1,a0
    adda.l targbase,a0
    move.l a0,tracead
    move.l #tracemg2,d1
    move.w #9,d0
    bdos
    bsr atol                ; Get trace end address.
    and.l $ffff,d1
    move.l d1,a0
    adda.l targbase,a0
    move.l a0,tracead
    move.w #10,d1          ; CRLF to end line.
    move.w #2,d0
    bdos
    move.w #13,d1
    move.w #2,d0
    bdos
    rts

*
*   OPEN file to be loaded, and load it into target
*   space if successful.
*
loadcom link a6,#0        ; Mark stack frame.
    movem.l d2-d3/a2-a4,-(sp)
    move.l 12(a6),a0       ; Get the address of the base page.
    lea.l $5c(a0),a2       ; Get FCB address.
    move.b #'C',9(a2)      ; mash filename to .COM
    move.b #'O',10(a2)
    move.b #'M',11(a2)
    move.l a2,d1
    move.w #15,d0
    bdos
    cmpi.w #255,d0         ; OPEN file.
    beq openerr           ; ERROR?

filelod    move.l pseudopc,d2 ; Start loading at $0100 in target.
    move.l d2,d1          ; Set DMA address.
    move.w #26,d0
    bdos
    move.l a2,d1
    move.w #20,d0          ; Read file until EOF.
    bdos
    tst d0
    bne basepg
    add.l #128,d2
    bra filelod

basepg    lea.l $80(targbase),a2 ; Set up the target's base page.
    move.l a2,d1          ; Start with default DMA address.
    move.l a2,dmaaddr
    move.w #26,d0
    bdos
    lea.l $38(a0),a2
    lea.l $5c(targbase),a3 ; Copy host's 2nd FCB to target's 1st FCB.
    move.w #36,d0
    fcbloop move.b (a2)+,(a3)+
    dbra d0,fcbloop
    lea.l $80(a0),a2
    lea.l $80(targbase),a4
    lea.l $81(targbase),a3
    clr d0
    move.b d0,(a4)
    move.b (a2)+,d0        ; Grab command tail from host's buffer.
    cmp.b #520,(a2)+      ; Hack off ? .COM filename.
    dbeq d0,tail1
    bne loaded            ; If there's any tail left, then
    tail2    cmp.b #520,(a2)+ ; remove leading whitespace.
    dbne d0,tail2
    beq loaded
    dec.l a2
    subq #2,d0
    tail3    move.b (a2)+,(a3)+ ; Move the rest of the tail.
    inc.b (a4)
    dbra d0,tail3
    move.b #0,(a3)
    bra loaded

openerr    move.l #opnmsg,d1 ; Can't open file.
    move.w #9,d0
    bdos
    clr d0

loaded    movem.l (sp)+,d2-d3/a2-a4
    unik a6                ; Transtor.
    rts

page
*****
*
*   BIOS and BDOS service request handler.
*
*****

service    moveq #0,d0      ; Handle BIOS/BDOS service request
    move.b (pseudopc)+,d0 ; of form HLT DB opcode.
    bne biosfn            ; BDOS or BIOS?

biosfn     moveq #0,d1
    move.b regc(regs),d0   ; Get BDOS function number.
    move.w regd(regs),d1   ; Get argument.
    cmp #31,d0            ; Can't do Disk Parm Hdr function
    beq badbdos           ; or ALLOC vector fn.
    cmp #27,d0
    bne okbdos

badbdos    move.l #ilgbdos,d1
    move.w #9,d0
    bdos
    bsr dump
    bra quitprg

okbdos     cmp #9,d0        ; Translate target address to real address.
    blt noconv
    cmp #14,d0
    beq noconv
    cmp #32,d0
    beq noconv
    cmp #37,d0
    beq noconv
    add.l targbase,d1
    noconv    cmp #26,d0      ; Save last known DMA address
    bne notdma            ; (in case of OPEN processing).
    notdma    move.l d1,dmaaddr
    move.b #0,fcflag      ; Separate FCB type requests
    cmp #15,d0            ; from the rest of the swine.
    blt notfcg            ; (Assume not, at first).
    cmp #24,d0
    blt fcb
    cmp #30,d0
    beq fcb
    cmp #33,d0
    blt notfcg
    cmp #37,d0
    blt fcb
    cmp #40,d0
    beq fcb
    bra notfcg

fcb        page
    swap d2
    move.w #35,d2          ; Move the FCB to host working buf,
    move.l d1,a0
    move.l a1,-(sp)
    fcb1     lea.l fcbstor,a1
    move.b (a0)+,(a1)+
    dbra d2,fcb1
    move.l (sp)+,a1
    lea.l fcbstor,a0      ; and swap the random record bytes
    move.b 33(a0),d2      ; to make them match the 68000's.
    move.b 35(a0),33(a0)
    move.b d2,35(a0)
    swap d2
    move.b #1,fcflag      ; Set flag for proper recovery.
    move.l d1,-(sp)      ; (Gotta put the pig back in pen!)
    move.l a0,d1
    ifne trcdsk          ; Optional ^2 Register dump.
    ifne dmpdsk
    bsr dump
    endc
    endc
    cmp.w #15,d0
    bne notopen
    tst.b 12(a0)
    beq notopen
    beq openproc
    bra results

notopen:
    notopen:
    ifne trcdsk          ; Optional FCB trace.
    move.l d2,-(sp)
    move.b #' ',d2
    bsr fcbtrc1
    move.l (sp)+,d2
    endc

notfcg     cmp #6,d0        ; Not an FCB request. Is it
    bne notdcon          ; a direct console I/O function?
    cmp.b #5ff,d1        ; Yes, make host's look like target's.
    bne notdcon
    move.w #5fe,d1
    bdos
    tst d0
    beq results
    move.w #6,d0
    move.w #5ff,d1

notdcon    bdos            ; FINALLY! Do the translated function.
    results move.w d0,regh(regs)
    move.b d0,rega

```



```

move.b regh(regs),regb(regs)
tst.b fcbflag          ; Do we need to restore a FCB?
beq done
ifne trcdsk
bsr fcbtrc2
endc
lea.l fcbstor,a0        ; Restore the FCB to target, in proper
                        ; order.
swap d2
move.b 33(a0),d2
move.b 35(a0),33(a0)
move.b d2,35(a0)
move.l (sp)+,a0
move.l a1,-(sp)
lea.l fcbstor,a1
move.w #35,d2
fcb2 move.b (a1)+,(a0)+
      dbra d2,fcb2
      swap d2
      move.l (sp)+,a1
done  move.b rega,d0
      and.w regconf,d0
      move.b 0(flagptr,d0.w),regf
      rts

openproc:
~openproc:
      ifne trcdsk          ; Optional FCB trace.
      swap d2
      move.b #' ',d2
      bsr fcbtrc1
      swap d2
      bsr fcbtrc2a
      endc

      move.b 33(a0),-(sp)    ; Save away RR fields!
      move.b 34(a0),-(sp)
      move.b 35(a0),-(sp)
      movem.l d0-d2,-(sp)
      moveq #0,d2
      move.b 12(a0),d2      ; Save desired extent.
      clr.b 12(a0)

```

```

bsr fcbddos              ; Do BDOS (with opt. tracing).
tst.b d0
bmi badopen              ; No seek if not good OPEN.
asl.l #7,d2              ; Make EXTENT # into record offset.
moveq #0,d0
move.b 32(a0),d0
bclr #7,d0
add.l d2,d0              ; Add onto CR to make abs record #.
move.w d0,34(a0)         ; Put into FCB.
swap d0
move.b d0,33(a0)
move.l #junkbuf,d1       ; Set DMA addr elsewhere for Rand Seek.
move.w #26,d0
bdos
movem.l (sp)+,d0-d2
move.w #33,d0            ; Random READ (SEEK) desired extent.
bsr fcbddos              ; Do BDOS (with opt. tracing).
clr d0                   ; (OPEN) must always be successful because
                        ; of the way CP/M-80 & CP/M-68K differ
                        ; on OPENing non-zero extents.
                        ; Restore the proper DMA address.
*
*
movem.l d0-d1,-(sp)
move.w #26,d0
move.l dmaaddr,d1
bdos
movem.l (sp)+,d0-d1
restore move.b (sp)+,35(a0) ; Restore RR fields.
        move.b (sp)+,34(a0)
        move.b (sp)+,33(a0)
        rts

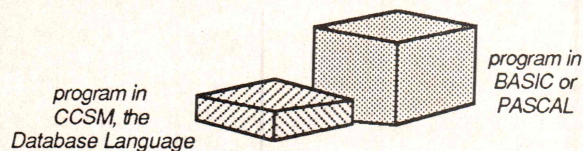
badopen movem.l (sp)+,d0-d2
        bra restore

fcbddos:
~fcbddos:
      ifne trcdsk          ; BDOS call with optional FCB trace.
      move.l d2,-(sp)
      move.b #'+',d2
      bsr fcbtrc1

```

(Continued on next page)

Write 1/3 the code with CCSM, the Database Language!



Only \$59.95 for IBM PC!
Only \$89.95 for Macmumps!

Dramatically increase your programming productivity with CCSM, the Database Language™. If you're now writing programs in BASIC or PASCAL, you'll be writing about 1/3 the code with this compact, productive language.

CCSM, the Database Language™, developed by COMP Computing, is a proprietary version of ANSI Standard MUMPS.

MUMPS as a language has been refined and developed for the past 20 years, and is used in corporate America, and by countless thousands around the world, who must manage large, complicated filing systems. With CCSM, the Database Language™, you won't be restricted with different "dialects", as you are with BASIC. CCSM, the Database Language™, will transport directly from your microcomputer, to a mini or mainframe WITHOUT MODIFICATION!

A "Database Language" Is The Answer

CCSM, the Database Language™, is "type-less", so you can use variables "on the fly", without the tyranny of type declarations, such as found in even the "souped-up" versions of PASCAL. The system transparently converts words and numbers to their real function, so you can add up inventory, or multiply commissions, or check on customers. Variables are written in plain English, such as "vendor", "cost", or "product". There are no line number conventions, but you may use plain English labels for reference to subroutines. CCSM, the Database Language™, doesn't make you decide when to use "sequential" files, or "random" files. Every record is stored in a "B-tree" file. So, you have virtually instant

access by simply choosing the appropriate search "key". With our "B-tree" files, you have variable length keys, and data, and you never have to restructure or re-sort your files. The "B-tree" record structure serves as temporary AND permanent storage, so you never have to worry about DIMensioning an array, because the records ARE the array. They're stored on disk for your immediate use. So, unlike BASIC, you never have to "GET" or "PUT", to store, retrieve or manipulate your data.

All The Goodies...Graphics, Too!

CCSM, the Database Language™, utilizes virtual memory, so programs or local variable sizes can expand to the size of the disk. CCSM, the Database Language™, has built-in error checking, full-screen editor, and sophisticated buffer pooling for faster performance, than RAM-disk alternatives. 8087 and BCD support, too.

Easy-to-Learn With Complete Documentation

You get a 250-page manual, with every chapter designed to lead you step-by-step through the language. The section "Introduction to MUMPS" is especially helpful, in making an easy-to-learn language, even easier. Through Mar. 31, 1986, you can order "The Cookbook of MUMPS", with its own disk of routines, and utilities, for only \$15.95 (reg. \$24.95). For an additional \$49.95, you can order the graphics disk, to create fancy charts and graphs. There is a multi-user package (up to 15) available, for \$450. All are non-copy-protected.

Stop Kludging, and Start Computing!

Database solutions are as close as your phone. Call Guidance Software now, and get CCSM, the Database Language™. You'll need an IBM PC, IBM compatible or Apple Macintosh with at least 128K.

1-800-257-8052 in Texas, (713) 529-2576
 AMEX, VISA or MC
Guidance Software, PO Box 5362, Kingwood, Texas 77325

CCSM, the Database Language™ is \$59.95 (\$89.95 for Macintosh), including the disk and 250 page manual. Get both language and utilities disks, along with "The Cookbook of MUMPS" for \$75.90 (\$105.90 for Macintosh). For graphics disk, add \$49.95. Include \$3.00 for shipping and handling. Texas residents add 6 1/8% tax.

IBM PC is a trademark of International Business Machines; Macintosh is a trademark of Apple Computer; CCSM, the Database Language & Macmumps are trademarks of COMP Computing.

8080 SIMULATOR

LISTING ONE (Listing Continued, text begins on page 76)

```

        move.l (sp)+,d2
        endc
        bdos
        ifne trcdsk
        bsr fcbtrc2
        endc
        rts

biosfn  cmp #1,d0                ; Handle Bios calls.
        beq quitprg
        cmp #5f,d0              ; List Status is ok.
        beq gudbios
        cmp #7,d0
        bge badbios             ; Don't allow disk functions!
gudbios clr.w d1
        move.b regc(regs),d1
        movem.l d2-d7/a0-a6,-(sp)
        bios
        movem.l (sp)+,d2-d7/a0-a6
        move.b d0,rega
        rts

badbios move.b d0,-(sp)          ; Flag illegal BIOS call
        move.l #biosmsg,d1      ; and spill guts.
        move.w #9,d0
        bdos
        move.b (sp)+,d1
        bsr pbyte
        move.l #biosmg2,d1
        move.w #9,d0
        bdos
        bsr dump

quitprg move.l (sp)+,d0          ; Trash return address and
        rts                    ; quit simulation.

        page
*****
*                               *
*       FCB Tracing support routines.       *
*                               *
*****

        ifne trcdsk
fcbtrc1 movem.l d0-d2/a0,-(sp)  ; Dump to printer each FCB usage
        move.b #9,d1            ; in format FN #, Disk, Name (ASCII)
        bsr lpchar              ; and the rest, all in hex but the
        move.w d0,d1            ; name field. Print the returned
        bsr lpbyte              ; value after the FCB.
        move.b d2,d1            ; Char in D2 is printed after FN #.
        bsr lpchar
        bsr lpspace
        bsr lpspace
        move.b (a0)+,d1
        bsr lpbyte
        bsr lpspace
        move.w #10,d2
fcbtr1  move.b (a0)+,d1          ; Print Name field...
        bsr lpchar
        dbra d2,fcbtr1
        bsr lpspace
        move.w #3,d2
fcbtr2  move.b (a0)+,d1          ; Ex .. Rc
        bsr lpbyte
        bsr lpspace
        dbra d2,fcbtr2
        bsr lpspace
        bsr lpspace
        lea.l 16(a0),a0          ; Skip d0..dn field.
        move.w #3,d2
fcbtr3  move.b (a0)+,d1          ; CR .. R2
        bsr lpbyte
        bsr lpspace
        dbra d2,fcbtr3
        bsr lpspace
        bsr lpspace
        move.l dmaaddr,d1
        sub.l targbase,d1
        bsr lpword
        bsr lpspace
        movem.l (sp)+,d0-d2/a0
        rts

        page
fcbtrc2 movem.l d0-d1,-(sp)      ; Line termination of FCB trace.
        bsr lpspace
        bsr lpspace
        move.b d0,d1
        bsr lpbyte
fcbtr21 move.b #10,d1
        bsr lpchar
        move.b #13,d1
        bsr lpchar
        movem.l (sp)+,d0-d1
        rts

```

```

fcbtrc2a: movem.l d0-d1,-(sp)      ; Line termination if no result
        bra fcbtr21              ; is to be presented.
        endc

        page
*****
*                               *
*       Misc. service routines.             *
*       (Inelegant, but rarely used so they stand as is). *
*                               *
*****

pbyte  move.l #520010,d0         ; 2 nybbles, 24 bit shift first.
        bra pdigits
pword  move.l #540010,d0         ; 4 nybbles, 16 bit shift first.
        bra pdigits
paddr  move.l #560000,d0         ; 6 nybbles, 8 bit shift first.
        bra pdigits
plong  move.l #580000,d0         ; 8 nybbles, no shift first.
pdigits rol.l d0,d1              ; Do shift.
        bra pdigent
pdiglop swap d0                  ; Save nybble count.
        rol.l #4,d1              ; Print variable in d1.
        bsr ntoa
pdigent swap d0                  ; Get nybble count.
        dbra d0,pdiglop
        rts

ntoa   movem.l d0-d1,-(sp)       ; Nybble in d1 to ASCII, then output.
        and #5f,d1
        cmp #5a,d1
        blt ntoa2
        add.b #'A'-'9'-1,d1
ntoa2  add.b #'0',d1
        move.w #2,d0
        bdos
        movem.l (sp)+,d0-d1
        rts

pspace move.w #32,d1             ; Print a space.
        move.w #2,d0
        bdos
        rts

        page
*                               *
*       Line Printer versions of above       *
*                               *

lpbyte move.l #520010,d0         ; 2 nybbles, 24 bit shift first.
        bra lpdigits
lpword  move.l #540010,d0         ; 4 nybbles, 16 bit shift first.
        bra lpdigits
lpaddr  move.l #560000,d0         ; 6 nybbles, 8 bit shift first.
        bra lpdigits
lplong  move.l #580000,d0         ; 8 nybbles, no shift first.
lpdigits rol.l d0,d1              ; Do shift.
        bra lpdigent
lpdiglp swap d0                  ; Save nybble count.
        rol.l #4,d1              ; Print variable in d1.
        bsr lntoa
lpdgent swap d0                  ; Get nybble count.
        dbra d0,lpdiglp
        rts

lntoa  movem.l d0-d1,-(sp)       ; Nybble in d1 to ASCII, then output.
        and #5f,d1
        cmp #5a,d1
        blt lntoa2
        add.b #'A'-'9'-1,d1
lntoa2  add.b #'0',d1
lntoa3  move.w #5,d0
        bdos
        movem.l (sp)+,d0-d1
        rts

lpchar  movem.l d0-d1,-(sp)       ; Print a character.
        bra lntoa3

lpspace movem.l d0-d1,-(sp)       ; Print space.
        move.w #32,d1
        bra lntoa3

        page
*                               *
*       Remaining misc. service routines.    *
*                               *

lpstr  movem.l d0-d1,-(sp)       ; Print a null-terminated string.
lpstr1  move.b (a0)+,d1
        beq lpstr2
        bsr lpchar
        bra lpstr1

```



```

lpstr2 movem.l (sp)+,d0-d1
      rts

konin  move.w #1,d0      ; Console input for 'atol'.
      bdos
      rts

atol   moveq #0,d1        ; ASCII to long, stops on invalid hex char.
      clr d2              ; Returns long in d1, terminator char in d0.
atol1  bsr konin          ; d2=1 if any chars entered before terminator.
      cmp.b #54,d0
      blo atol2
      and #5F,d0          ; Mask to upper case.
atol2  cmpi.b #'0',d0      ; Check range (0..9,A..F).
      blo atolend
      cmpi.b #'F',d0
      bhi atolend
      cmpi.b #'9',d0
      bls atol3
      cmpi.b #'A',d0
      bhs atol3
      bra atolend
atol3  moveq #1,d2        ; Valid characters entered, flag it.
      sub.b #'0',d0
      cmp.b #59,d0
      bls atol4
      sub.b #'A'-'9'-1,d0
atol4  ext d0              ; To long.
      ext.l d0
      asl.l #4,d1          ; Tack it onto D1.
      add.l d0,d1
      bra atol1
atolend rts

      page
      data
*****
*
* Target processor's data registers.
* Fake FDOS.
*
*****

```

```

even
regop3 equ -9            ; Operand 1 for DAA storage.
regb   equ -8            ; Offsets from register base pointer for
regc   equ -7            ; 8080's pseudo-registers.
regd   equ -6            ; A & F are in Data Registers.
rege   equ -5            ; Pseudo-PC is kept in an Address Register.
regh   equ -4
regl   equ -3
regop1 equ -2            ; Operand 1 for DAA storage.
regop2 equ -1            ; " 2 " " " "

fcbstor ds.b 36          ; Host works FCB's out of here.
fcbflag ds.b 1           ; Flag says we used the FCB buffer.

even
tracesad ds.l 1          ; Trace start address.
traceead ds.l 1          ; Trace end address.
traceflg ds.w 1          ; Tracing enabled flag.

dmaaddr ds.l 1           ; DMA address storage.

page
fdos    dc.b $76,0,$C9    ; Fake BDOS for target system.
*       ; BIOS Jump Table
      dc.b $C3,$33,$FF    ; Wboot
      dc.b $C3,$36,$FF    ; Const
      dc.b $C3,$39,$FF    ; Conin
      dc.b $C3,$3C,$FF    ; Conout
      dc.b $C3,$3F,$FF    ; List
      dc.b $C3,$42,$FF    ; Punch
      dc.b $C3,$45,$FF    ; Reader
      dc.b $C3,$48,$FF    ; Home
      dc.b $C3,$4B,$FF    ; Seldsk
      dc.b $C3,$4E,$FF    ; Settrk
      dc.b $C3,$51,$FF    ; Setsec
      dc.b $C3,$54,$FF    ; Setdma
      dc.b $C3,$57,$FF    ; Read
      dc.b $C3,$5A,$FF    ; Write
      dc.b $C3,$5D,$FF    ; Listst
      dc.b $C3,$60,$FF    ; Sectran

```

(Continued on next page)

d/MULTI MULTIUSER dBASE for TurboDOS

TRUE File and Record Locking as easy as d-BASE-II. Unlimited users can perform the magic of dBASE in the program or interactive mode

- * TurboDOS 1.3 or 1.4
- * No Peeks or Pokes
- * System Date and Time Functions
- * Printspooler Controls up to 16 printers

Martian Technologies . . .

**.CREATeIng Multi-users from
Single-users around the world**

CALL FOR DETAILS

Martian Technologies

8348 Center Dr., Ste.-F, La Mesa, CA 92041
(619) 464-2924



Now available with 8087 Support! MTBASIC Basic Compiler

Features:

- | | |
|----------------------|---------------------|
| Multi-line functions | Multitasking |
| No runtime fee | Windowing |
| Handles interrupts | Interactive |
| Fast native code | Compiles in seconds |

MTBASIC is easy to use since you can write programs in an interactive environment and then compile them using only one command. MTBASIC has many advanced features like multitasking, random file access, formatted I/O, assembly language calls, and ROMable code.

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so any system can use windows), demonstration programs, and a comprehensive manual.

Ordering

MTBASIC is available for CP/M, MS-DOS, and PC-DOS systems for \$49.95. MTBASIC with 8087 support is available for MS-DOS for \$79.95. Shipping is \$3.50 (\$10.00 overseas). MD residents add 5% sales tax. MC, Visa, checks and COD accepted.



P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

LISTING ONE (Listing Continued, text begins on page 76)

End Listing One

```

*****
*
*      This file contains the target processor (8080) simulation
*      routines.
*
*****

*****

*
*      Opcode dispatch table. One longword entry per opcode of the
*      target (8080) processor, including illegals.
*
*****

globl op tabl, flags, nop80
xdef mloop, illegl, service, preED, outspec

diskio equ 0 ; Non-zero for special HDC/DMA support.

return equ 016,r ; JMP (return) is fast return to MLOOP.
pseudopc equ 015,r ; 8080's PC is register A5.
opptr equ 014,r ; Pointer to opcode dispatch table.
pseudosp equ 013,r ; 8080's SP is register A3.
flagptr equ 012,r ; Pointer to 8080's flag lookup table is A2.
targbase equ 011,r ; Pointer to 8080's address space is A1.
regs equ 011,r ; Base pointer to 8080's registers is A1.

regcon0e equ 7,r ; Register based constant #SE (for speed).
regcon0i equ 6,r ; Register based constant #S1.
regcon0f equ 5,r ; Register based constant #SF.

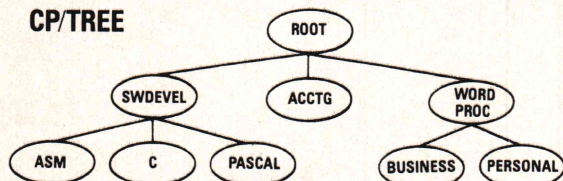
```

[illegible]

lxi b	move.b (pseudopc)+,regc(regs) move.b (pseudopc)+,regb(regs) jmp (return)	; 01 Lxi BC,nnnn	dcxb	dec.w regb(regs) jmp (return)	; 0B Dcx B
staxb	move.w regb(regs),d0 move.b rega,0(targbase,d0.1) jmp (return)	; 02 Stax B	inrc	inc.b regc(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 0C Inr C
inxb	inc.w regb(regs) jmp (return)	; 03 Inx B	dcrc	dec.b regc(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 0D Dcr C
inrb	inc.b regb(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 04 Inr B	mvic	move.b (pseudopc)+,regc(regs) jmp (return)	; 0E Mvi C
dcrb	dec.b regb(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 05 Dcr B	rrca	ror.b #1,rega bra docyf	; 0F Rrc
mvib	move.b (pseudopc)+,regb(regs) jmp (return)	; 06 Mvi b,nn	nop00	bra illegl	; 10 Illegal for 0000
rlca	rol.b #1,rega bcs rlc1 bclr #0,regf jmp (return)	; 07 Rlc	lxi d	move.b (pseudopc)+,rege(regs) move.b (pseudopc)+,regd(regs) jmp (return)	; 11 Lxi DE,nnnn
docyf	bclr #0,regf jmp (return)		staxd	move.w regd(regs),d0 move.b rega,0(targbase,d0.1) jmp (return)	; 12 Stax D
rlc1	bset #0,regf jmp (return)		inxd	inc.w regd(regs) jmp (return)	; 13 Inx D
nop08	bra illegl	; 08 Illegal for 0000			
dadb	move.w regb(regs),d0 add.w d0,regh(regs) bra docyf	; 09 Dad B			
ldaxb	move.w regb(regs),d0 move.b 0(targbase,d0.1),rega jmp (return)	; 0A Ldax B			

(Continued on next page)

CP/TREE



Tree-Structured Named Directories for CP/M 2.2

- Transforms user areas into Unix-like directories
- Provides Unix-like directory commands:
CD, MKDIR, PWD, RMDIR, TREE
- Includes a CCP replacement featuring:
 - Command and file search path. Use programs like WordStar from any directory!
 - Erase with query
 - Wildcard rename with query
- Provides output redirection to disk file
- Uses as little as 1/2 k RAM, never more than 2 1/4 k
- A must for hard disks
- Installs easily: requires no modifications to BDOS or BIOS
- Requires standard CP/M 2.2 (not 3.0 or Apple), Z80, 48k RAM

\$29.95 plus \$4.00 s&h

To order: Specify disk format (8" SSSD, NorthStar DD. Call for info on others.). MC, Visa, COD (add \$1.90), check (delays shipping 2 weeks). MA residents add 5% sales tax. POs not accepted.

Precise Electronics
P.O. Box 339
New Town Branch
Boston, MA 02258
tel: (617) 332-3977

Apple® Apple Computer. CP/M® Digital Research. WordStar® MicroPro International. Z80® Zilog. Unix® AT&T Technologies.

Published for Over 7 Years

68'

MICRO

JOURNAL

Serving The 68XXX
User Worldwide
The Bible of the
Serious User

\$2.95^{USA}

Australia A \$4.75 New Zealand NZ \$6.50
Singapore S \$6.45 Hong Kong H \$23.50
Malaysia M \$6.45 Sweden 30-SEK

68020
68010
68000
68008
6809

The ABC's of Languages
ADA-Basic-C

Subscribe NOW and Receive FREE Reprints Previous (1-8)
Continuous ADA Series offer exp. 2-15-86

Hardware-Software-Application Articles
New Product Releases-Announcements
Hints-Kinks-Fixes, etc.

Subscription Rates

U.S.A.: 1 Yr. \$24.50, 2 Yrs. \$42.50, 3 Yrs. \$64.50
*Foreign Surface: Add \$12.00 per Year to USA Price
*Foreign Airmail: Add \$48.00 per Year to USA Price
*Canada & Mexico: Add \$9.50 per Year to USA Price
*U.S. Currency or Check or Draft Drawn on USA Bank !!



(615) 842-6809

Telex 5106006630



5900 Cassandra Smith Rd.

Hixson Tn. 37343

Circle no. 213 on reader service card.

8080 SIMULATOR

LISTING TWO (Listing Continued, text begins on page 76)

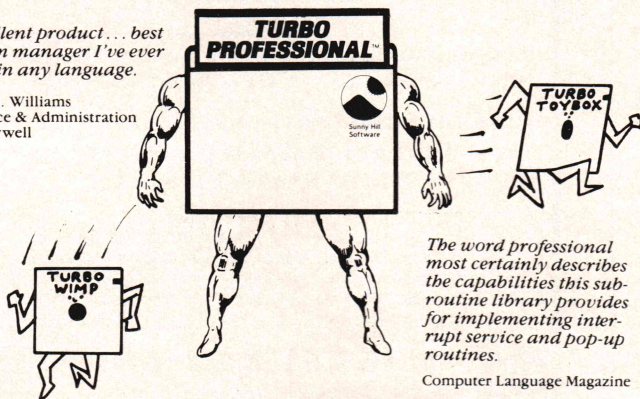
inrd	inc.b regd(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 14 Inr D	move.b (pseudopc)+,regh(regs) jmp (return)		
dcrd	dec.b regd(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 15 Dcr D	shld	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc move.l d0,a0 adda.l targbase,a0 move.b regl(regs),(a0)+ move.b regh(regs),(a0) jmp (return)	; 22 Shld addr
mvid	move.b (pseudopc)+,regd(regs) jmp (return)	; 16 Mvi D,nn	inxh	inc.w regh(regs) jmp (return)	; 23 Inx H
ral	roxr.b #1,regf roxl.b #1,rega roxl.b #1,regf jmp (return)	; 17 Ral	inrh	inc.b regh(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 24 Inr H
nop18	bra illegl	; 18 Illegal for 8080	dcrh	dec.b regh(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 25 Dcr H
dadd	move.w regd(regs),d0 add.w d0,regh(regs) bra docyf	; 19 Dad D	mvih	move.b (pseudopc)+,regh(regs) jmp (return)	; 26 Mvi H,nn
ldaxd	move.w regd(regs),d0 move.b 0(targbase,d0.l),rega jmp (return)	; 1A Ldax D	daa	move.b regop3(regs),d0 roxr.b d0 move.b regop2(regs),d0 move.b regop1(regs),d1 swap regcon0e move.b rega,regcon0e and.b regcon0f,regcon0e cmp.b #9,regcon0e bhi halfcy and.b regcon0f,d0 and.b regcon0f,d1 ori.b #5f0,d1 addx.b d0,d1 bcc nohalf	; 27 Daa
dcxd	dec.w regd(regs) jmp (return)	; 1B Dcx D	halfcy	add.b #6,rega bcs fullcy	
inre	inc.b rege(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 1C Inr E	nohalf	btst #0,regf bnz fullcy move.b rega,regcon0e and.b #5f0,regcon0e cmp.b #590,regcon0e bis nofull	
dcre	dec.b rege(regs) move sr,d0 and.w regcon0e,d0 and.w regcon01,regf or.b 0(flagptr,d0.w),regf jmp (return)	; 1D Dcr E			
mvie	move.b (pseudopc)+,rege(regs) jmp (return)	; 1E Mvi E,nn			
rar	roxr.b #1,regf roxr.b #1,rega roxl.b #1,regf jmp (return)	; 1F Rar			
nop20	bra illegl	; 20 Illegal for 8080			
lxih	move.b (pseudopc)+,regl(regs) jmp (return)	; 21 Lxi H,nnnn			

(Listings to be Continued next month)

ATTENTION TURBO PASCAL PROGRAMMERS

Excellent product... best screen manager I've ever seen in any language.

Don A. Williams
Finance & Administration
Honeywell



The word professional most certainly describes the capabilities this sub-routine library provides for implementing interrupt service and pop-up routines.

Computer Language Magazine

Get the tools that have no competition.

TURBO PROFESSIONAL's easy to use routines let you...

- write safe, "sidekickable" routines that can use DOS
- execute DOS commands from Turbo
- create super-fast windowed displays
- service interrupts without assembler
- make your own keyboard enhancers
- allocate memory from DOS
- print concurrently with DOS 3.x.

Complete with 109 + routines, manual, source code to Super Macs keyboard enhancer, and many example programs.

Only \$49.95 + \$5.00 shipping & handling

Visa, MasterCard ok.

Order now!
(206) 367-0650



Sunny Hill Software
13732 Midvale North Suite 206
Seattle, Washington 98133

Requires Turbo for compatibles. Turbo Pascal & Sidekick Trademark Borland Intl.

Circle no. 172 on reader service card.

Fatten Your Mac for \$5.00

Thanks to Macintosh owners everywhere, *Dr. Dobb's* January 1985 issue #99 was a runaway best-seller.

Now, due to popular demand, the Doctor has reprinted the sought-after **Fatten Your Mac** article from the sold-out January issue. The article explains how you can pack a full 512K of memory into your system, and save half the cost by performing the upgrade yourself.

To order: Enclose \$5.00 for each copy with this coupon and send to:

Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Outside U.S., add \$2.00 per copy for shipping & handling.

Please send me _____ copies of **Fatten Your Mac**. **ALL REPRINT ORDERS MUST BE PREPAID.**

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 6-9 weeks for delivery.

Offer expires Dec. 31, 1985

311 G

U S Software

Design Engineering Software —
Embedded Application Support

68000 USX™

Multi-Tasking Operating System Executive

- Real-time and ROMable
- Dynamic task management
- Priority task scheduling
- Memory management
- Intertask communication
- Sample debug system included
- User defined configurations
- Less than 3K bytes
- Delivered in source assembly for ease of configuring and customizing functions for specific job applications

USX product line also supports the 8086/8088, 8051 and 8096 microprocessor families.

United States Software Corporation
5470 N.W. Innisbrook Place • Portland, Oregon 97229 • (503) 645-5043
International Telex 4993875 (U S Software)

Circle no. 214 on reader service card.

C Cross Compiler 68000/08/10/20

Features:

- Full, Standard C
- Easy to Use Compiler Options
- Complete User Documentation
- Global Code Optimization
- Optional Register Allocation Via Coloring
- ROMable and Reentrant Code
- Comprehensive Royalty Free Run-time Library
- Floating Point Library Routines
- Intermix MCC68K C with ASM68K Assembly Language or Microtec PAS68K Pascal
- Optional Assembly Language Listing Intermixed with MCC68K C Source Line Number
- Symbolic Debug Capability

The Microtec MCC68K C Cross Compiler is a complete implementation of the 'C' programming language as defined in The C Programming Language by Kernighan and Ritchie with extensions.

MCC68K emits highly optimized assembly language code for the Microtec ASM68K Motorola compatible assembler.

The Microtec MCC68K package includes the compiler, relocatable macro assembler, linking loader, run-time library, and comprehensive user's guide.

Host computers include: DEC VAX, PDP-11, DG MV-Series, DG Eclipse, Apollo, UNIX Systems, IBM PC, Data General/ONE, HP 150, DEC Rainbow, and others.

We're **Functional** and **Fast** and **Serious** about our products. We've been providing flexible and economical solutions for software developers since 1974.

Beginning with product concept, through development, quality assurance, and post-sales support - **Quality, Compatibility and Service** are the differences which set Microtec Research apart from others.

If you're a serious software developer, shopping for software development tools, call or write today for more information:

800-551-5554,
In CA call (408) 733-2919.

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088

MICROTEC®
RESEARCH

ANNOUNCING! DR. DOBB'S COMPLETE TOOLBOX OF

*Dr. Dobb's Journal,
the most respected source of
technical software information available,
brings you this collection
of powerful programming tools for C.*

New, from M&T Publishing and
Brady Communications . . .

Dr. Dobb's Toolbook for C

Item #005

A comprehensive library of valuable C code.

Many of Dr. Dobb's most popular articles on C from sold-out issues are updated and reprinted in this unique reference, along with new C programming tools, **THE TOOLBOX**, contain's C-compilers, line editors, assemblers, text processing programs, and more! Dr. Dobb's Journal offers **The Toolbook** in a special hardbound edition for only \$29.95. You'll find:

- J.E. Hendrix's famous *Small C Compiler v. 2* and *A New Library for Small C* (also available on disk)
- Never before published: Hendrix's new *Small-Mac: An Assembler for Small C* and *Small Tools: Programs for Text Processing* (both available on disk)
- Ron Cain's original *A Small-C Compiler for the 8080's*
- Plus many useful programming tools in C

Also from M&T Publishing and
Brady Communication—

The Small-C Handbook

Item #006 or #006A

The Small-C Handbook by James Hendrix is a valuable resource of information about the Small-C Compiler. In addition to descriptions of the language and the compiler, **The Handbook** also contains the entire source listings of the compiler and its library of arithmetic and logical routines. A perfect companion to the Hendrix Small-C compiler offered by Dr. Dobb's on disk, **The Handbook** even tells how to use the compiler to generate new versions of itself. All this is in **The Handbook** for only \$17.95, Item #006; only \$22.95 for **The Handbook** with the MS/PC DOS Handbook Addendum, Item #006A.

While both **The Toolbook** and **The Handbook** provide documentation for the Small-C Compiler on disk, **The Handbook** provides more complete documentation and is available with an Addendum for MS/PC DOS versions.



DR. DOBB'S C TOOLS ON DISK

To complement The Toolbook, Dr. Dobbs' also offers the following programs on disk. Source code is included, and except where indicated, both CP/M and MS/PC-DOS versions are available.

Small C Compiler Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software published in Dr. Dobbs' 10-year history. Like a home study course in compiler design, the Small-C Compiler and The Small-C Handbook provide all you need to learn how compilers are constructed, as well as teaching the C language at its most fundamental level. The Small-C Compiler is available for \$19.95 in both CP/M and MS/PC DOS versions.

Both The Toolbook and The Small-C Handbook provide documentation for the compiler; however, The Handbook provides more complete documentation for both versions, and the MS/PC DOS Small-C Handbook Addendum is recommended in addition to The Handbook for MS or PC DOS specific documentation.

Small-Mac: An Assembler for Small-C Item #012A

Small-Mac is a macro assembler designed to stress simplicity, portability, adaptability, and educational value. The package features simplified macro facility, C-language expression operators, descriptive error messages, object file visibility, and an external table. Included programs are: macro assembler, linkage editor, load-and-go loader, library manager, CPU relocation utility, and dump program. This program is available with documentation for the Small Mac Manual for \$29.95. For CP/M systems only.

Small Tools: Programs for Text Processing Item #010A

This package consists of programs designed to perform specific functions on text files, including: editing, formatting, sorting, merging, listing, printing, searching, changing, translating, copying and concatenating, encrypting, and decrypting, replacing spaces with tabs and tabs with spaces, counting characters, words, or lines, and selecting printer fonts. Source code only is included. This program is available with documentation in the Small Tools Manual for \$29.95. Both CP/M and MS/PC DOS versions available.

Special Holiday Packages — 20% Off!

Now, for almost 20% off the individual product prices, you can order a complete set of Dr. Dobbs' C programming tools for your CP/M or MS/PC DOS system!

CP/M C Package Only \$99.95 Item #005A

Ordered individually, these C tools sell for over \$120! Order the CP/M C Package now and you'll receive Dr. Dobbs' Toolbook, The Small-C Handbook, The Small Tools Processor on disk, along with documentation in The Small Tools Manual, The Small-Mac Assembler on disk, and The Small-Mac Manual—all for only \$99.95!

MS/PC DOS C Package Only \$82.95 Item #005B

Ordered individually, these tools sell for over \$100! Order now and you'll receive: Dr. Dobbs' Toolbook, The Small-C Handbook with the MS/PC DOS Addendum, The Small Tools Processor on disk, along with documentation in the Small Tools Manual—all for only \$82.95!

Books	Dr. Dobbs' Toolbook			
	Small-C Handbook	Check Format		
	Small C Handbook with MS/PC DOS Addendum	CP/M	MS/PC DOS	
Disks	Small-C Compiler (Toolbook or Handbook recommended for documentation)			\$19.95
	Small Tools Text Processor with Small Tools Manual			\$29.95
	Small Mac Assembler with Small Mac Manual (for CP/M only)			\$29.95
				\$82.95
SPECIAL HOLIDAY PACKAGES	CP/M C Package			
	MS/PC DOS C Package			
Subtotal				%
Tax				
Shipping				
TOTAL				

CA residents add applicable sales tax. Add \$1.75 per item for shipping in U.S., \$4.25 outside U.S.

For CP/M system disks only, please specify one of the following formats:
☐ Kaypro ☐ Apple ☐ Zenith Z-100 DS/DD ☐ Osborne ☐ 8" SS/SD
 Inquire about other formats.

PAYMENT MUST ACCOMPANY YOUR ORDER.
☐ Check Enclosed
☐ Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # _____ Exp. _____
 Signature _____ State _____ Zip _____
 Name _____
 Address _____
 (Please use street address)
 City _____

Please return this coupon along with your payment to: Dr. Dobbs' Journal, 2464 Embarcadero Way, Palo Alto, CA 94303. For credit card orders, call toll-free: 1-800-528-6050 Ext. 4001. Refer to item numbers.
 Please allow 4-8 weeks for delivery.

3111B

LISTING ONE (Text begins on page 118)

```

;
; DUMMY SEGMENT TO BE GROUPED WITH HIMEM, WHICH
; IS AN EMPTY SEGMENT AT THE END OF THE USER'S
; PROGRAM
;
HIMEM GROUP HIDATA
HIDATA SEGMENT COMMON 'HIMEM'
FLARB DW 0
HIDATA ENDS
;
;
; 'MAIN' IS THE PROGRAM'S ENTRY POINT,
; JUST ABOVE THE PSP
;
EXTRN MAIN:FAR
;
ARC CODE SEGMENT PUBLIC 'CODE'
;
    ASSUME CS:ARC CODE
    PUBLIC FREMEM
;
FREMEM PROC FAR
;
    PUSH BP                ; STANDARD ENTRY STUFF
    MOV BP,SP              ;
;
; GET THE ADDRESS OF HIMEM, THE TOP OF THE
; USER'S PROGRAM, STICK IT IN USER'S VARIABLE
;
    MOV AX,SEG HIDATA      ; GET TOP SEGMENT
    LES BX,18[BP]          ; SEND IT TO THE USER
    MOV ES:[BX],AX        ;
;
;
; FIND OUT HOW MUCH MEMORY IS AVAILABLE ABOVE
; HIMEM BY REQUESTING A LOT OF MEMORY.
; THE CALL TO FUNCTION 4AH NEEDS THE SEGMENT

```

```

; ADDRESS OF THE PROGRAM'S PSP. GET IT BY
; SUBTRACTING 10 PARAGRAPHS FROM THE SEGMENT
; OF THE PROGRAM'S ENTRY POINT, WHICH HAS THE
; LABEL 'MAIN'.
;
    MOV AX,SEG MAIN        ; GET ENTRY SEGMENT
    SUB AX,10H             ; ADJUST TO GET PSP
    MOV ES,AX              ; PUT VALUE INTO ES
    MOV BX,-1D             ; GET ALL OF MEMORY
    MOV AX,4A00H           ; DOS FUNCTION CODE
    INT 21H                ; DO IT
;
; THE NUMBER OF AVAILABLE PARAGRAPHS IS IN BX,
; RETURN IT TO USER.
;
    LES DI,14[BP]          ; STORE IT IN
    MOV ES:[DI],BX         ; USER'S VARIABLE
;
;
; TRY TO ALLOCATE THE PARAGRAPHS REQUESTED.
; A VALUE OF -1 MEANS DON'T ALLOCATE ANY.
;
    LES DI,10[BP]          ; GET THE USER'S
    MOV BX,ES:[DI]         ; VARIABLE
    CMP BX,-1D             ; SHOULD WE ALLOCATE?
    JE OK                  ; NO, BAIL OUT
    MOV AX,SEG MAIN        ; GET ENTRY POINT
    SUB AX,10H             ; ADJUST TO GET PSP
    MOV ES,AX              ; PUT IT IN ES
    MOV AX,4A00H           ; DOS FUNCTION CODE
    INT 21H                ; DO IT
    JNA ERR                ; CHECK FOR ERRORS
OK:  XOR AX,AX             ; NONE, CLEAR FLAG
ERR: LES BX,6[BP]          ; STORE ERROR CODE
    MOV ES:[BX],AX         ; IN USER'S VARIABLE
;
; THAT'S IT
;
    MOV SP,BP              ; STANDARD EXIT STUFF
    POP BP
    RET 16
;
FREMEM ENDP
;
ARC CODE ENDS
END

```

End Listing One

Now available for the computer experimenter!

COMPUTER CONNOISSEUR'S DELIGHT!

NOW BE IN CONTROL WITH YOUR COMPUTER — THE ONLY PUBLICATION OF ITS KIND WRITTEN FOR THE USER. DISCOVER THE SECRETS AND LEARN THE VERSATILITY OF MODERN COMPUTER COMMAND AND CONTROL CONCEPTS. EXPERIMENT WITH COMPUTER AND TELEPHONE SYSTEMS, INTERFACE THEM, LEARN HOW THEY WORK, WHAT THEY DO... AND HOW TO GET THEM TO WORK FOR YOU! A COMPLETE TELEPHONE ENGINEERING COURSE IS INCLUDED IN MONTHLY CHAPTERS, BRINGING YOU THROUGH STEP, CROSSBAR, ESS, BUBBLE, AND ATOMIC SWITCHING SYSTEMS! EXCLUSIVE COVERAGE IN BIOLOGICAL COMPUTING SYSTEMS, TOO! COMPUTERS AND TELEPHONES ARE THE FUTURE. THIS PUBLICATION IS AN ABSOLUTE MUST FOR EVERYONE INTERESTED.

UNPUBLISHED MATERIAL

WIT

COMICS

DIRECTORY LISTING NETWORKS

ACCESS CODES

The one you've all been waiting for

NOW AVAILABLE — Learn how to repair telephones and telephone systems, how they work, in monthly installments with the magazine for you.

Computel™

PUBLISHED MONTHLY

ONE YEAR SUBSCRIPTION \$14.00
(SAMPLE COPY \$2.00)
SUBSCRIPTION & 2 PROGRAMS \$20.00

COMPUTEL—the complete SOURCE for everyone. You can now do the things you've only heard about, right in the privacy of your own home. Indispensable reference to phreaks and hackers. Learn how to get all kinds of computer programs FREE. Get the inside story of big business systems—their quirks and flaws—and remain up to date with vital occurrences within the computer industry. Computel is a publication designed for everyone who has an intense curiosity of computer systems, containing a wealth of hard to find information, codes, and numbers. Published monthly.

Computel Publishing Society

6354 VAN NUYS BL., #161-A / VAN NUYS, CA 91401

Circle no. 225 on reader service card.

LISTING TWO

; Michael Barr's 32-bit Square Root Routine

```

; Call with CX:BX = argument
; Returns BX = result
;

```

```

sqrt    proc    near                ; CX:BX = argument
        push    ax                  ; save other registers
        push    dx
        push    di
        jcxz    sqrt3              ; prepare first try
        mov     dx,cx
        mov     di,-1
sqrt1:  shl     dx,1
        shl     dx,1
        jc      sqrt2
        shr     di,1
        jmp     sqrt1
sqrt2:  mov     dx,cx                ; restore argument
        mov     ax,bx
        cmp     dx,di              ; prevent overflow
        jae     sqrt4
        div     di
        cmp     ax,di              ; comp quotient and divisor
        jae     sqrt4
        add     di,ax              ; average them
        rcr     di,1
        jmp     sqrt2
sqrt3:  mov     dx,bx
        mov     di,0ffh
        or      bx,bx              ; lower half zero?
        jnz     sqrt1              ; no, jump
sqrt4:  mov     bx,di              ; return result in BX
        pop     dx
        pop     ax
sqrt    endp

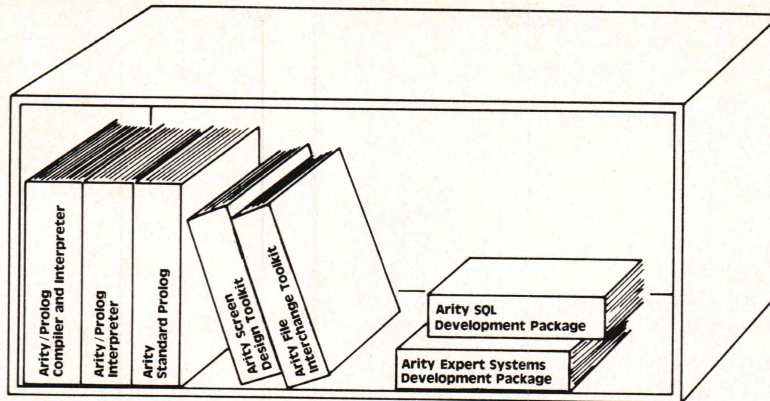
```

End Listings

We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.



arity



Why your next generation of products should use our 5th generation tools.

A rity's integrated family of programming tools allows you to combine software written in Aryity/Prolog,TM the best of the fifth generation languages, with Aryity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Aryity/Prolog to build expert systems using the Aryity Expert System Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Aryity/Prolog lets you build advanced technology into your vertical applications package.

And more . . .

That's not the whole story. Aryity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Aryity/Prolog. The most powerful of its kind on a PC.

Quality First. Then Price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. And we offer generous educational and quantity discounts on every one of our products.

If we are new to you, we do not ask that you trust us. You have to try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone — you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Aryity products.

Please complete this form to place your order and/or request detailed information

Name

Shipping Address

City State Zip

Telephone

	Quantity	Info Only
Aryity/Prolog Compiler and Interpreter V4	\$795.00	<input type="text"/>
Aryity/Prolog Interpreter	\$350.00	<input type="text"/>
Aryity Standard Prolog	\$ 95.00	<input type="text"/>
Aryity SQL Development Package	\$295.00	<input type="text"/>
Aryity Expert System Development Package	\$295.00	<input type="text"/>
Aryity Screen Design Toolkit	\$ 49.95	<input type="text"/>
Aryity File Interchange Toolkit	\$ 49.95	<input type="text"/>

Total Amount
(MA residents add 5% sales tax) \$

(These prices include shipping to all U.S. cities)

Payment: ☐ Check ☐ PO ☐ AMEX ☐ VISA ☐ MC

Card No. Expiration Date

Signature

arity
Aryity Corporation
358 Baker Avenue Concord, MA 01742

M66

16-BIT SOFTWARE TOOLBOX

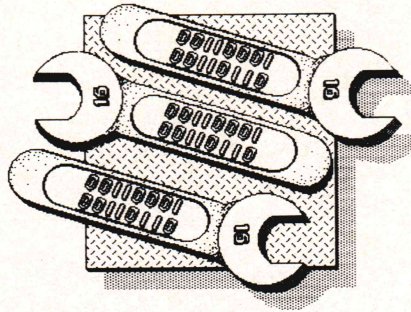
COLUMNS

Trojan Horse Programs

One of the more ominous developments in the last few months is the appearance of so-called Trojan horse programs on some public bulletin board systems. These have appeared under such names as EGABTR, VDIR, and SYSUTIL. They are usually accompanied by scanty documentation that bills them as super disk directories or something similar, but their actual effect is to trash your system by formatting the hard disk, erasing the file allocation table, or writing random garbage into files. I put the people who create and upload such programs to public BBSS in the same category as terrorists.

Along similar lines, some villains have taken advantage of their knowledge of the public-domain PC remote BBS to upload programs that purport to paint a pretty picture on the screen or do something else cute but actually copy the password file or other vital system files to (unprotected) files under another name. The villain calls back later and downloads the unprotected files, thereby gaining access to privileged files and messages.

As the computer terrorists become more clever, we can expect the appearance of subtle sabotage programs that copy themselves to hidden files on the hard disk or attach themselves to the bootable operating system and don't do their damage until much later. For example, I can envision a hard disk destruction program that would wait until it saw that you had not run Backup for a week! The nature of the damage could also be so subtle that it would drive you crazy, such as simply changing a bit in a random sector of



the hard disk every few days.

These developments have the potential to damage or destroy the proliferation of public-domain software on bulletin boards, which would be very sad. BBS operators will not want to take the risk of being held liable for disasters due to Trojan horse programs. On the LMI RBBS, we are immediately adopting a policy of deleting all programs that are not uploaded in the form of, or at least accompanied by, source code as a clear-text ASCII file. I'd like to hear comments from readers on this subject, especially those who have been victims or who can provide actual samples of these Trojan horse programs.

EXEC Calls and FORTRAN

Robert Sypek of a firm called Argis in Hudson, Massachusetts, writes: "While attempting to write a program that would execute a user or system task from a user program, I ran into many of the problems described in past issues about using the DOS 2.x EXEC function call. My problems were compounded by the fact that Microsoft's languages (FORTRAN, Pascal, C) set up their own data and stack segments, and the code segment is that of the user program or subroutine, not the segment of the root program. This means that the program segment prefix (PSP) is inaccessible from any routine called by a user program. All is not lost, however, as I believe I have found a method of retrieving the necessary information to allow an EXEC function to be called.

"The method depends on two

pieces of information that may be gleaned from the Microsoft user manuals: the compiler defines a null segment called *HIMEM* that is placed at the end of all other segments when the user program is loaded, and the compiler defines a symbol called *MAIN* that is located directly after the PSP. Both the segment and the symbol have the *PUBLIC* attribute, making them accessible to the user.

"The segment of the PSP may be found by subtracting 10H from the segment of *MAIN*, yielding the value of the ES register needed for the call to *EXEC*. The value of *HIMEM* can then be used to calculate the size of the user program and the segment of the next available paragraph of memory."

Mr. Sypek enclosed an assembly-language subroutine that we are printing as Listing One (page 116). The routine is invoked in the form:

```
CALL FREMEM(MEMPTR, MEMAVL,  
            MEMALL, ERR)
```

where *MEMPTR* is the segment value of the next available paragraph, *MEMAVL* is the number of paragraphs available, and *MEMALL* is a user-defined variable specifying the amount of memory above the program to leave allocated to the user. (A value of -1 leaves all memory allocated to the program.) An error code of 0 indicates that the function was successful.

Lightweight Reading

Those who think the mainframe mentality is gone forever should read Martin Healy's article "Toward a Viable OS for the PC" (*Datamation*, September 1985). At first I took this article for a practical joke because it contains so many distortions of the history and current state of the art in microcomputers, sideways slams at a variety of targets, and outrageous gobbledygook (example: "PC Network could be the answer, but it won't share data, only the file").

by Ray Duncan

The programs published in this month's column are available for downloading from the Laboratory Microsystems RBBS at (213) 306-3530 (300 baud or 1,200 bps).

The author asserts that "there are in fact two leading real operating systems for micros, Unix and Concurrent DOS from Digital Research." He goes on to say, "The new Version 4.1 [of Concurrent DOS] is the idealized multitasking PC DOS, which due to its maturity should eliminate any Microsoft version (PC DOS 4.0?) thereof. That leaves Microsoft to concentrate on its Unix-like system, Xenix." I am sure the folks at Microsoft will be very relieved to learn from Mr. Healy that they no longer have to waste all that effort on maintaining MS DOS, can take MS DOS 4.0 out of testing and toss it in the trash can, and turn their attention to other matters.

Square Roots

Michael Barr of the Department of Mathematics at McGill University sent us his 8086 assembly-language subroutine for square roots (Listing Two, page 116). He writes: "This routine gives the correct floored square root for any 32-bit number (considered as unsigned). It is also faster than the bit-at-a-time algorithms you have put into *DDJ*."

"Apropos that last statement, there seems to be a discussion between people who believe that Newton's method is always the best way to do a square root and others who believe equally fervently that the bit-at-a-time method is always faster. Common sense would dictate that they are both wrong. I strongly suspect that Newton's method is faster if and only if you have an on-chip (or co-processor) division of the relevant size. In particular, to do a 32-bit square root, you need a 32-bit by 16-bit division. This much I have tested; Newton's method is just about twice as fast (about 330 msec, compared to about 650 msec for the bit-at-a-time). What I haven't tested (I can't face the thought of programming them) are 64-bit square roots. But there is every reason to believe that it will be faster to do the bit-at-a-time square root than to code division and then use that for Newton's method."

Big DOS

The new wave of PCs, based on the 80286 microprocessor, are still limited to 1 megabyte of direct memory addressing because they run MS DOS or its clones in 8086 emulation mode

(called Real Mode by Intel). The ability of the 80286 to address 16 megabytes of RAM in "Protected Virtual Memory Mode" is currently supported only by the 80286 Xenix and iRMX-286 operating systems, neither of which is likely to become very popular due to their incompatibility with WordStar, dBASE III, and Lotus 1-2-3. The average user's exploitation of the full capabilities of the PC/AT and other such machines awaits the arrival of an operating system that runs in Protected Mode and can execute the popular

MS DOS applications unaltered.

Such an operating system has been the subject of much industry rumor and speculation. Digital Research has been talking about "Concurrent DOS-286" for some time now, even announcing the operating system's "immediate availability" at a press conference in New York about six months ago. (See "Concurrent DOS-286: Available Now," *Intel Speak Softly Quarterly*, vol. 2, no. 2, Second Quarter 1985.) But lately DRI has been backpedaling a bit and now admits

Mystic Pascal

Fastest Compiler on Earth—\$64!

Mystic Pascal compiles at 100,000 to over 1,000,000 lines per minute! How? It takes a short cut called *incremental compilation*. Compared to earlier Pascals, the effective speed is astronomical. Give the Compile command for a 1000 line program, and you probably can't lift your finger from the keyboard before the compiler flashes—DONE!

640K of storage not 64K. Are you fed up with being forced to shoehorn your programs into 64K? You won't need mystical powers to run your program in the full 640K that we allow—code, data and stack.

Interactive Pascal. You can enter Pascal statements directly and see the results instantly. It works like a Basic interpreter but it's a true compiler!

Optimized 8086 Code. Mystic Pascal produces true 8086 object code. The TWO code optimizers run in the background so they don't slow you down. Thanks to another breakthrough, our software floating point arithmetic runs faster than Intel says is possible on the 8088/8086 chips.

Real Multi-Tasking Pascal. Advanced programmers may write truly concurrent Pascal programs by simply starting Pascal procedures. Up to 100 concurrent procedures can communicate by passing messages through queues.

ISO Standard Pascal. Educators in particular need a truly Standard Pascal for their students. And learning is made easier by the Help Windows which describe Pascal.

Mystic Canyon Software

P.O. Box 1010

Pecos, New Mexico 87552

Place your order by phone today—
(505) 757-6344 or mail the coupon.

Requires an IBM Personal Computer or true compatible with 256K. Not copy protected.

Rush me the complete Mystic Pascal System with diskette and manual!

Name

Address

City State Zip

☐ Check/Money Order ☐ Visa ☐ Mastercard ☐ COD

Price is \$64 total.

Outside US & Canada add \$15. Payment must be in US funds on a US bank.

Card Exp.

Signature

Circle **no. 79** on reader service card.



Continuing the Tradition DR. DOBB'S JOURNAL BOUND VOLUMES

Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computer construction, and the technical history of personal computing. Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI.

Vol. 2 1977

These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of material centering microcomputers and a great deal of material centering around the Intel 6800, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL. Articles are about Lawrence Livermore Lab's BASIC, Alpha Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot, & Turtle Graphics, many utilities.

Vol. 3 1978

This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today. Languages covered in depth were SAM76, Pilot, Pascal, and Lisp; in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages, innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/8085, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference. Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe.

Complete your reference library. Buy the entire set of Dr. Dobb's Journal Bound Volumes 1-8, for \$195.00. That's \$34.00 off the combined individual prices—a savings of almost 15%!

Vol. 5 1980

Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published here in full. Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler, Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler.

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features. Highlights: Information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi delivered his famous review of JRT Pascal and wrote the first serious comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

Vol. 8 1983

DDJ turns pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. The second half of Jim Hendrix's Small C editor. A microcomputer subset of the Defense Department's official programming language, Ada. C and Fortran and 68000 software. Because the magazine increased in size in 1983, this volume is bigger and better than ever.

Dr. Dobb's Journals from 1976 through 1983, Bound Volumes 1-8, for \$195.00. That's \$34.00 off the combined individual prices—a savings of almost 15%!

Each book in this series contains a year's worth of Dr. Dobb's Journal monthly issues, reprinted in full and combined in one comprehensive volume. The Bound Volumes will fill the gaps in your Dr. Dobb's collection and complete your reference library

YES! ☐ Please send me the following Volumes of Dr. Dobb's Journal.

Payment must accompany your order.

Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

I enclose ☐ Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____

(please, no P.O. Boxes)

City _____ State _____ Zip _____

Mail to Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

3111C

Allow 3-6 weeks for delivery.

Vol. 1	x	\$26.75	=	_____
Vol. 2	x	\$27.75	=	_____
Vol. 3	x	\$27.75	=	_____
Vol. 4	x	\$27.75	=	_____
Vol. 5	x	\$27.75	=	_____
Vol. 6	x	\$27.75	=	_____
Vol. 7	x	\$30.75	=	_____
Vol. 8	x	\$31.75	=	_____
All 8	x	\$195.00	=	_____

Sub-total \$ _____

California residents add applicable sales tax _____%

Postage & Handling Must be Included with order.

Please add \$2.25 per book in U.S. (\$5.25 each surface mail outside U.S. Foreign Airmail rates available on request.)

TOTAL \$ _____

that 80286 Concurrent DOS will probably never exist in the form originally advertised. Of course, the company is blaming its problems on "defects" in the 80286 design. (When in doubt, fall back on the classic programmer's defense: "It must be a hardware problem.")

Microsoft Corp. is also known to be working on a Protected Mode, upward-compatible version of PC DOS (already dubbed Big DOS by the trade rags and assumed to be Version 5.0). But Microsoft has been very close-mouthed about these efforts, presumably wanting to make sure it can pull it off before committing itself to such a product in public. Too bad DRI wasn't that careful!

Ross Nelson, who previously worked on the 80286 team at Intel, took the time to write to us with some musings on the future of DOS and Protected Mode. He says, "With regard to the 286 in the marketplace, it seems to me that unless a comparatively low-cost system (PC/II?) is introduced, there won't be a lot of work done that will take advantage of Protected Mode (PM). The installed base of AT users vs. the number of PC-compatible users will make it economically unfeasible. Once people do start working with PM, they will encounter some interesting problems. The great virtue of PM is that no task in the system should be able to corrupt another task (assuming the operating system is stable). As a software engineer, I applaud this philosophy, and I believe that this use of the 286 should be encouraged. Realistically, however, it is clear that there will be a transition period in which PM will only be used to gain access to the larger memory space.

Nelson wrote: "As far as I can tell, there are only two ways of switching back to Real Mode when you are in Protected Mode, and only one of them is feasible with the standard 80286 part. This is essentially the method IBM has chosen [in the VDISK driver supplied with PC DOS 3.x . . . RD], which is to place enough state information to restart your process in a 'safe' location and RESET the processor. The other method requires a special 'bond-out' part (which Intel uses in

its I2-ICE). By activating a special pin on the bond-out chip, you can issue special instructions that dump and restore the internal state of the machine, including the Machine Status Word. Systems built with the bond-out chip could easily be toggled between Real Mode and Protected Mode.

"Whether or not a DOS 5.0 or Big DOS can be successful in emulating the current PC system architecture on a 286 will depend on how freely the implementors translate the word *compatible*. I do not believe that 100

percent compatibility can be accomplished without unreasonable overhead. I suspect that even partial compatibility will have large memory requirements. It would not surprise me to see a 512K operating system with an additional 32K required on a per-task basis. Here is how I believe some of the problems that come up can be handled:

"Video—This has quite a few simple solutions: (a) trap each initial write to the display segment, and replace the offending segment register with

The First Idea-Processor For Programmers.

FirstTime™

Has features no other editor has.



- Fast program entry through single keystroke statement generators.
- Fast editing through syntax oriented cursor movements.
- Dramatically reduced debugging time through immediate syntax checking.
- Fast development through unique programmer oriented features.
- Automatic program formatter.

New Updated Features. Text editing with 2 windows • User definable formatting

- Update includes files within FirstTime • When reading a file, cursor moves to next syntax error
- Command DOS from within FirstTime

FirstTime is a True Syntax Directed Editor.

As the world's most advanced syntax-directed editor, FirstTime lets you work with ideas, by taking care of the low-level syntax details of your program. For example, you can generate complete statement skeletons with one keystroke. Move the cursor from one procedure to the next with one keystroke. Type in code, and it's instantly formatted (you specify the rules). Type an error, and FirstTime warns you immediately. You can continue working if you wish. Later, you can use the search-for-error command to find the error and fix it.

FirstTime Has Thorough Error Checking.

FirstTime not only checks your syntax, but also semantics. FirstTime identifies:

- Undefined variables, types and constants.
- Assignment statements with type mismatches.
- Errors in include files and macro expansions.

To Order Call: (201) 741-8188 or write:

SPRUCE TECHNOLOGY CORPORATION



P.O. Box 7948
Shrewsbury, NJ 07701

FirstTime is a trademark of Spruce Technology Corporation • MS DOS is a trademark of Microsoft Corporation • IBM is a trademark of International Business Machines Inc. • Turbo Pascal is a trademark of Borland International • dBase III is a trademark of Ashton-Tate

FirstTime Lets You Work With Ideas.

The *Zoom* command gives you a top down view of your program logic.

The *View macro* command shows the expansions of a C macro while in the editor.

The *View include file* command instantly shows you the contents of an include file.

The *Transform* command allows you to change a statement to another similar statement, for example, a *FOR* to an equivalent *WHILE*.

The *Search for next error* command allows you to find errors throughout your program.

The *Cursor movement* commands let you traverse your program by logical elements, not just characters.

FirstTime Works With Existing Files.

FirstTime works with standard ASCII files, so you can edit any existing source files.

FirstTime for Turbo Pascal	\$ 74.95
FirstTime for dBase III	\$125.00
FirstTime for MS-Pascal	\$245.00
FirstTime for C	\$295.00

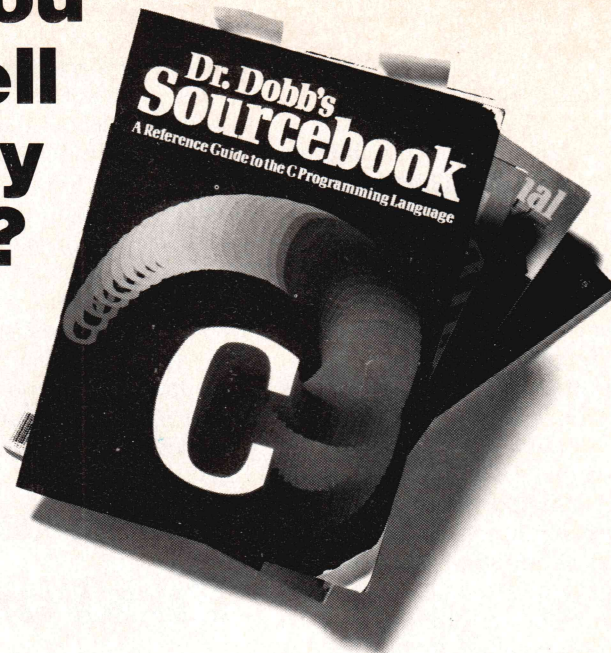
Requires MS DOS, 256k (384 recommended) & 5.25" drive

In Germany, Austria and Switzerland contact:
Markt & Technik Software Verlag
Munchen, W. Germany
(089) 4613-0

Circle no. 164 on reader service card.

Who Says You Can't Tell A Book By Its Cover?

Dr. Dobb's Sourcebook: A Reference Guide for the C Programming Language



For years, serious programmers have relied on Dr. Dobb's Journal for the technical tools of their trade. Now, Dr. Dobb's presents the definitive programmers guide to the who, what, where, when and why of C, the leading language among software developers. This comprehensive guide to new information, products and services specific to C will be your most often-used reference!

In this valuable guide you'll find:

- An extensive directory of hardware and software services—including classes and seminars, C programming opportunities, and on-line services
- A bibliography with over 300 listings of available articles and books on C
- A comprehensive C product listing—including C compilers, graphics modules, utilities, editors and development systems, and more!
- And much more practical C programming information

At only \$7.95, no C programmer can afford to be without this unique reference.

To order by credit card, call toll free: 1-800-528-6050 ext.4001 . Ask for item 004 or mail this coupon, along with payment to **Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303**

PAYMENT MUST ACCOMPANY YOUR ORDER

_____ I enclose check/money order
_____ Please charge my _____ VISA _____ M/C _____ American Express
Card # _____ Exp. Date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

Please allow 6 to 12 weeks for delivery

Please send me _____ copies of **Dr. Dobb's Sourcebook**

at \$7.95 each = _____

+ Shipping & Handling = _____

(Must be included with order. Please add \$1.50 per book in U.S. \$3.25 each surface mail outside U.S. Foreign airmail rates available on request.)

TOTAL = _____

3111D

(Continued from page 121)

an OS-created descriptor; (b) use the TopView solution, that is, require applications to issue a software interrupt to get the address of their own 'local' display buffer; (c) trap every display write and deal with it on a case-by-case basis. Solution (c) is the most compatible but the poorest performer; (a) is almost as compatible and would substantially increase performance. Solution (b) is the best and would not cause a great deal of incompatibility, especially with programs that use installable device drivers.

"Communications—Here we have no simple solutions. I would expect that all but the most primitive communications applications would have to be rewritten. Anything more complicated than Int 14H calls should be declared incompatible and rewritten to fit the new OS.

"EXE files—Here, the OS must limit the 'free-memory' size to 64K of code and 64K of data. Programs that need more memory should be forced to [dynamically] allocate it. Because of the large number of programs that indiscriminately load segment registers, however, an OS might want to trap segment load faults and attempt to map them into the 8086-style physical addressing scheme.

"COM files—These programs are the most likely to be incompatible, because they often load and stay resident and have only one segment (CS). Because executable segments are never writable, an OS would incur severe performance penalties trying to simulate the standard DOS mode of operation here. Some heuristics would work for some programs, such as loading DS, SS, and ES with a writable alias of the CS descriptor, but any program that did segment register arithmetic would then yield incorrect results.

"Obviously, there are hundreds of similar problems. . . . The question is, what solutions will the marketplace accept? It seems that a Protected Mode 286 operating system will have to contain a large measure of DOS/IBM PC compatibility to prevent the 'software gap' problem faced by the Macintosh, Amiga, etc. Unfortunately, this means a lower-performance,

aesthetically unpleasing solution. I would welcome a radically different, optimized system, but only IBM has the clout to pull it off, and there would still be two important factors: IBM would have to want to do it, and it would have to do it right."

Nothing but the Best for My Little Girl

Richard Gaulden, vice president of sales and marketing for EDCOM, Inc., wrote me a letter describing his company as a "national supplier of enrichment materials to school and library systems." His sales pitch

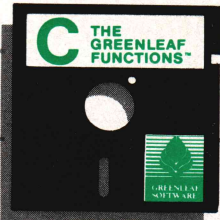
wound up with the statements: "EDCOM can free your distribution channels by marketing software that becomes outdated by new releases. The educational market does not have the need for constant upgrades, especially when new versions are more costly." Kind of gives you a warm, secure feeling about your kids' education in computer literacy, doesn't it?

DDJ

(Listing begins on page 116)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 6.**



THE GREENLEAF FUNCTIONS™

The GREENLEAF FUNCTIONS GENERAL LIBRARY has over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and systems interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions**. Simply the finest C library (and the most extensive). All ready for you.


THE GREENLEAF FUNCTIONS™
The Library of C Functions that probably has just what you need . . . **TODAY!**

- already has what you're working to re-invent
- already has over 200 functions for the IBM PC, XT, AT, and compatibles
- already complete . . . already tested . . . on the shelf
- already has demo programs and source code
- already compatible with all popular compilers
- already supports all memory models, DOS 1.1, 2.0, 2.1
- already optimized (parts in assembler) for speed and density
- already in use by thousands of customers worldwide
- already available from stock (your dealer probably has it)
- It's called the **GREENLEAF FUNCTIONS.**

The Library of C Functions is Waiting for You

Specify compiler when ordering. Add \$7.00 for UPS second-day air (or \$5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check or P.O. In stock, shipped same day.

■ General Libraries	\$185	For Information: 214-446-8641
■ Comm Library	\$185	
■ C/C86 Compiler	\$349	
■ Lattice C	\$395	Prices are subject to change without notice.
■ Mark Williams	\$475	



GREENLEAF SOFTWARE
2101 HICKORY DR.
CARROLLTON, TX 75006

PROFESSIONAL PROGRAMMER

Kitchen-Table Entrepreneur

The purpose of this department is to provide information of use to professional programmers. This first installment addresses itself to the independent software developer, romantically and perhaps not altogether inaccurately envisioned as working on a VAX by day and by night designing the compiler of tomorrow on an XT on the kitchen table. We point here to a number of books that might be of use to the kitchen-table entrepreneur.

Copyright

Read books on copyright with some caution. Both copyright law itself and the interpretations of it in court have changed in their application to software publishing. Don't assume that you need not copyright your software if you favor a Freeware/Shareware approach to distribution.

Remer, Daniel. *Legal Care for Your Software*. Berkeley, Calif.: Nolo Press, 1984.

Read this for discussion of all the legal issues: work-for-hire agreements, non-disclosure agreements, contracts, license agreements. The author covers the differences among and nuances of copyright, trade-secret protection, patent, and trademark registration. The book explains your legal liabilities and how international copyright works. It has a number of sample forms, such as a beta test site agreement form.

Salone, M.J. *How to Copyright Software*. Berkeley, Calif.: Nolo Press, 1984.

Read it for deeper discussion of copyright issues. What happens when the first 500 disks go out with no copyright notice on the disk or in the code? What can you do if your copyright is infringed? Hundreds of examples are provided.

Documentation

Read something on documentation and realize how important it is. A good manual is, after all, the best copy protection available. Many programmers cut their own throats by releasing good software with illiterate, poorly designed documentation. Besides creating an aura of amateurishness for your product, such a manual will hide the capabilities of your product from users—and however experienced a programmer you may be, the odds are you're an amateur in producing documentation.

Houghton-Alico, Doann. *Creating Computer Software User Guides*. New York: McGraw-Hill, 1985.

Read it to jog your memory about some of the techniques available for communicating about your software—maybe what your compiler documentation needs is a good pie chart. Maybe not. Use the book to get an idea of what professional documentation writers have to do, though their tasks are somewhat different from yours, of course.

Stephan, Peter M. *Writing User-usable Manuals*. Salt Lake City: Wredco Press,

1984.

Read it as an example of decent low-cost documentation. The author has won awards for his documentation, and although the book may not tell you anything you don't already know, it shows that a manual need not be typeset and perfect bound.

Strunk, William, and White, E.B., *The Elements of Style*. New York: Macmillan, 1972.

Read it. Whether you write the documentation or hire writers, your words are likely to see print somewhere. There is no more concise guide to keeping your foot out of your semiliterate mouth than this book.

Markets

Read these if you're really an independent software developer and don't want to distribute your product(s) yourself. These books mainly list software distributors.

Amato, Francis. *Guide to Computer Magazines*. Dallas: Steve Davis, 1985.

Read it for a quick idea of the editorial focus, circulation, and audience of selected computer magazines. This can be useful for promotional and advertising purposes, and some of the magazines are software markets in themselves, publishing programs in their pages and/or on disks.

Hoffman, Roger. *The Complete Software Marketplace*. New York: Warner Books, 1984.

Read it for many reasons. One reason is to get an overview of the things you

need to know in running a small software business. The book has lists of distributors, including mail-order companies, electronic distributors, and magazines. There are also lists of disk duplication services, PR firms, market research firms, trade associations and shows, seminars, and conferences. There are case studies and a useful section on freebies for the independent author.

McGehee, Brad M., ed. *1986 Programmer's Market*. Cincinnati: Writer's Digest Books, 1985.

Read it for publishers who use "free-lance" material—even the terminology reflects this book's heritage. It was put together by people who view the independent software developer as another kind of author. If that description fits you, this book may also.

Software Design

Read this to remind yourself of the basic principles and also to learn a vocabulary for communicating the principles to others.

General Electric. *Software Engineering Handbook*. New York: McGraw-Hill, 1986.

Read it for one approach to the management of large software projects when your design team actually becomes a team. Also read its bibliography for sources on design methods: Nassi/Sneiderman, Warnier, Yourdon, and Jackson methodologies; data-flow driven design.

DDJ

BOOKSHELF™

**Fast, compact, high quality,
easy-to-use CP/M system**

Series 100

**Priced from
\$895.00
10MB System
Only \$1645.00**



Features

- Ready-to-use professional CP/M computer system
- Works with any RS232C ASCII terminal (not included)
- Network available
- Compact 7.3 x 6.5 x 10.5 inches, 12.5 pounds, all-metal construction
- Powerful and Versatile:
 - Based on Little Board/PLUS single-board computer
 - Two RS232 serial ports
 - One Centronics printer port
 - One or two 400 or 800 KB floppy drives
 - 10-MB internal hard disk drive option
- Comprehensive Software Included:
 - Enhanced CP/M operating system with ZCPR3
 - Word processing, spreadsheet, relational database, spelling checker, and data encrypt/decrypt (T/MAKER III)
 - Operator-friendly shells; Menu, Friendly™
 - Read/write and format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
 - Menu-based system customization
- Expandable:
 - Floppy expansion to four drives
 - Hard disk expansion to 60 megabytes
 - SCSI/PLUS™ multi-master I/O expansion bus

AMPRO
COMPUTERS, INCORPORATED

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0930 • TELEX 4940302

T/MAKER III is a trademark of T/Maker Company.
IBM is a registered trademark of International Business Machines.
Z80A is a registered trademark of Zilog, Inc.
CP/M is a registered trademark of Digital Research.

Circle **no. 158** on reader service card.

Presenting SMK, the

SEIDL MAKE UTILITY™

SMK is a UNIX like make utility for MS-DOS, but without the cryptic commands or limitations. When changes are made to any program module, SMK will issue only those commands *necessary and sufficient* to rebuild the system. SMK is a professional, full-featured make utility.

Advanced SMK features include:

- Super fast analysis of dependency files. Proprietary dependency algorithm analyzes *all* dependencies before rebuilding any files.
- SMK understands complicated dependencies involving nested include files, source and object code libraries.
- High-level dependency definition language makes setting up dependencies easy. Supports parameterized macros, local variables, constants, include files, command line parameters, in-line and block comments.
- Works on any MS-DOS (PC-DOS) computer and with most compilers, assemblers and linkers.
- Full pathname and directory support.
- Typeset user's manual and excellent error diagnostics make SMK easy to learn and use.

Price: \$99.95

(add \$3.50 postage & handling)

SEIDL COMPUTER ENGINEERING

1163 E. Ogden Ave., Suite 705-171
Naperville, IL 60540
(312) 983-5477

(SMK, UNIX, MS-DOS, PC-DOS are all trademarks.)

Circle **no. 114** on reader service card.

NEON™ TURN ON THE FULL POWER OF YOUR MACINTOSH.™

Hidden within your Mac is the programming power, flexibility, and speed to match your imagination. Neon is your key to this object-oriented world. Based on the same design philosophy as the Mac itself, Neon lets you create and command objects — program modules that you build, perfect, and add to your personal collection of tools. In this Smalltalk-like environment, you wield the power to quickly assemble and test ideas, tuning as you go for maximum speed and efficiency. □ Neon works in 128K or dynamically expands for the fattest Mac possible. It is easy to work with from the very beginning. Additional help is provided through the comprehensive manual by Danny Goodman. □ Created by Kriya Systems, Inc. for the development of our Typing Tutor III™ program, Neon is your answer for professional software development. With Neon, there are no licensing fees. Ever. For the Macintosh developer, Neon is the best choice.



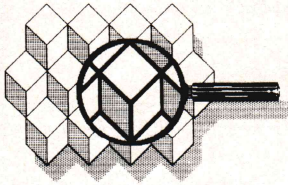
To order your copy of Neon, call 1-800-34KRIYA now with Visa or MasterCard, or send your check to Kriya Systems, Inc., Six Export Drive, Sterling, VA 22170. The price is \$155, including shipping and handling.

Neon and Typing Tutor III are trademarks of Kriya Systems, Inc. Apple Macintosh is a trademark of Apple Computers, Inc.

Circle **no. 100** on reader service card.



OF INTEREST



Peak Electronics has announced a 68K coprocessor board that, it claims, you can plug into any (IEEE-696-1983) S-100 system running CP/M-2.2 and have CP/M-68K running in minutes with no hardware or software modifications. The 68K8-CP is a 10-MHz processor card with an MC68008 (8-bit version of the 68000), up to 512K RAM, up to 128K EPROM, two 38.4 kilobaud serial ports, and an 8-bit parallel printer port. The 68K8-CP doesn't replace your current CPU card but runs in conjunction with it, so you can jump back and forth between operating systems. Peak says that the board will support Concurrent DOS 68K when it's released.

Speaking of Concurrent DOS 68K, maybe DRI is finally going to exploit the head start it had in 68K development software: The company was expected to introduce its 68K developer's kit at Comdex in November. With Concurrent, DRI has two operating systems for the 68000; three if you count the GEM windowing environment. Concurrent DOS 68K has a CP/M front end that will run most CP/M-68K programs, and DRI is promising GEM support for subsequent versions of Concurrent DOS 68K.

Speaking of GEM, last month we discussed it and other windowing environments and, because of space limitations, mentioned only some of the

most visible. We don't feel right about giving all that attention to the big boys and never mentioning two windowing products we have had good experience with: Desqview and dWindow. It was, of course, not only Digital Research, IBM, and Microsoft that went windowing. Quarterdeck's Desqview is an impressive competitor to TopView, which bundled with AST's Rampage expanded memory board, lets users run up to nine programs concurrently in memory. Also, the success of the concept of windows is exemplified by its employment in a nonoperating system application in the product dWindow. For years, Ashton-Tate's dBASE II set the standard for austerity in user interfaces: You can't get much simpler than a single-period prompt. Liberty Bell Publishing's dWindow does a dazzling cathedral window treatment on dBase II (and now dBase III) that makes it look like an entirely different product.

Speaking of entirely different products, Answer Software has announced an 80286 emulator for developers of 80286-based software. The ICD286 consists of a card for the host system (which must be in the IBM PC/XT/AT family or a compatible), a Buffer Box that plugs into the 80286 slot in the target system, and a symbolic debugger. It allows uploading and downloading of code and data, hardware and software breakpoints, single-stepping, and full-speed emulation up to 10-MHz clock rates.

And speaking of 80286 development, American

ADO has introduced an 80286 board for Multibus systems. The SOL C286-01 (no relation to the Processor Technology Sol computer of story and song) is being manufactured in 6-, 8-, and 10-MHz versions and has up to 512K RAM, a Centronics printer interface, and two 8- or 16-bit SBX bus I/O connectors. Then there's the ET-286Plus, a 10-MHz AT-compatible single-board computer that uses the new 1-Mb dynamic RAMs and allows 4 megabytes on-board. ATS International was expected to show it at Comdex.

As long as we're speaking of the 286 and operating systems, we should mention Locus Computing's Multisystem Merge. This product allows simultaneous, transparent execution of Unix and MS DOS on the same machine according to the company. You can set several Unix tasks to work in the background while you run a DOS application in the foreground. This is the system AT&T is using on its 6300+ Unix/DOS computer. Locus developed some of the technology in the system while writing PC-Interface, a product that links DOS computers to a host Unix machine.

DRI, which we were speaking of a few paragraphs back, is of course not the only developer of operating system software for the 68K, as two recent announcements prove. U.S. Software has announced a real-time multitasking system for embedded applications using the 68K. It's ROMable, requires 3K of code space, and is called USX68K. Integrated Busi-

ness Computers has ported TheOS-16 (formerly Oasis) to its line of 68010 computers and was expected to be showing a beta version at Comdex.

Speaking of beta-test versions, Tall Tree Systems has begun shipping beta versions of its J laser-printer interface to software companies it deems closest to achieving compatible products. The interface is a PC/XT/AT card that works with the JRAM 2-megabyte memory board; it's designed to spend memory to buy print speed and typeface flexibility for laser printers. It transfers bit-mapped images directly from RAM to the print mechanism and is supposed to provide unlimited type fonts with full graphics capabilities at 300 dots per inch in eight seconds.

Speaking of *mucho* megabytes, Reference Technology has announced a device that lets you attach up to eight of its optical disk drives, for more than 4 gigabytes of storage on a (sturdy, large) desktop. The device is PC/XT/AT or compatible compatible.

Speaking of product announcements, Speech Technology is now selling the cross-assemblers it developed in the process of designing and manufacturing electronic devices to aid the blind (its real business). The MS DOS cross-assemblers for the 8048 and 6502 were written in C and support a subset of C preprocessor commands, macros, three object file formats, and features to support PROM programming. They sell for \$30 each or \$75 for

both with source code and are distributed with no restriction on noncommercial copying.

Speaking of copying, we are watching with interest SoftKlone's fortunes with a product it cheerfully describes as a clone of Microstuf's Crosstalk XVI data-communications package. SoftKlone's Mirror was designed to the precise visual specs of Crosstalk, and SoftKlone presents itself as introducing a new idea—mirror-image software at a lower price than the mirrored product and perhaps with added capabilities. Rather than competing by producing a better or cheaper product, the notion here is to produce the same product better or cheaper.

Reference Map

Peak Electronics, P.O. Box 700112, San Jose, CA 95170; (408) 253-5108. Reader Service Number 16.
Digital Research, P.O. Box DRI, Monterey, CA 93942; (408) 649-3896. Reader Service Number 17.
Liberty Bell Publishing, 618 N.W. Glisan, Ste. 203, Portland, OR 97209; (503) 221-1806. Reader Service Number 18.
Quarterdeck Office Systems, 1918 Main St., Ste. 240, Santa Monica, CA 90405; (213) 392-9851. Reader Service Number 19.
Answer Software, 20863 Stevens Creek Blvd., Cupertino, CA 95014; (408) 253-7515. Reader Service Number 20.
American ADO, 1840 West 186th St., Ste. 200, Tor-

rance, CA 90504; (213) 532-5010. Reader Service Number 21.

ATS International, 2105 Luna Rd., Ste. 330, Carrollton, TX 75006; (214) 247-5151. Reader Service Number 22.

Locus Computing Corp., 3330 Ocean Park Blvd., Santa Monica, CA 90405; (213) 452-2435. Reader Service Number 23.

U S Software, 5470 N.W. Innisbrook Pl., Portland, OR 97229; (503) 645-5043. Reader Service Number 24.

Integrated Business Computers, 21621 Nordhoff St., Chatsworth, CA 91311; (818) 882-9007. Reader Service Number 25.

Tall Tree Systems, 1120 San Antonio Rd., Palo Alto, CA 94303; (415) 964-1980. Reader Service Number 26.

Reference Technology, 1832 North 55th St., Boulder, CO 80301; (303) 449-4157. Reader Service Number 27.

Speech Technology Inc., 16321 176th Ave. N.E., Woodinville, WA 98072; (206) 483-5150. Reader Service Number 28.

SoftKlone, 1210 East Park Ave., Tallahassee, FL 32301; (904) 878-8564. Reader Service Number 29.

DDJ

THE HAMMER Software Tools in C

"I have already saved weeks of coding . . . thank you for providing such a useful tool . . ." - G.T.

Let **The HAMMER Library** of over 150 routines save you valuable development time and effort:

LIBRARY FUNCTIONS

• Multi-level 123-like MENUS

• DATA ENTRY

- MULTI-FIELD mode for Full-Screen data entry
- Single-Field mode for individual fields
- Data Verification
- Full Editing within each field
- Strings, dates, and fixed decimal numbers
- "Option" fields force user to pick from a given set

• SCREEN MANAGEMENT

- cursor positioning
- display boxes & tables
- full attribute control
- scrolling and clearing

- Date/time/string conversions
- BIOS access/pattern matching/and more

UTILITIES

- HARC - complete Source File Archiver
- HAMCC - compile designated source modules residing **WITHIN an archive file** under any of the supported compilers and optionally place resulting object modules in a library.

SUPPORTED C COMPILERS:

Microsoft C 3.00 • C1-C86 • Mark Williams C86
DeSmet C • Lattice

INCLUDES source code and manual

\$195 plus shipping
VISA/MC accepted

O.E.S. SYSTEMS

1906 Brushcliff Rd. • Pittsburgh, PA 15221 • 412/243-7365

Looking for the right tool for the job?

REACH FOR THE HAMMER

Circle no. 137 on reader service card.

COHERENT for the AT

A fast UNIX-compatible O/S for networking, communications and process control

- ▶ Comes with unique disk partitioning capability that allows storage of DOS or other O/S on same disk as COHERENT. Special kernel enhancements make it easy to configure COHERENT to individual system requirements.
- ▶ Includes complete COHERENT development environment with C compiler, screen editor and source code control system. Available with UNIX System V compatible support for inter-process communications.
- ▶ Total object code compatibility between the IBM AT, XT and PC.
- ▶ Optional support available for front-end communications processor functions, high resolution graphics, image capture, mouse and other hardware. Flexible turnkey and runtime licensing available. Professional technical support provided.

To discuss your system requirements call: (604) 294-6201

INETCO Systems Ltd.

2457 Beta Ave., Burnaby, B.C., Canada V5C 5N1

COHERENT is a trademark of Mark Williams Co. UNIX is a trademark of Bell Laboratories. IBM AT, XT and PC are trademarks of International Business Machines Corp.

Circle no. 166 on reader service card.

GROWING OLD?

...waiting
for C programs to
compile and link?



Use C-terp
the complete C interpreter
This is the product you've been
waiting (and waiting) for!

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

C-terp Features

- Full K&R C (no compromises)
- Complete built-in screen editor-- no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, etc. etc. For the ultimate in customization, editor source is available for a slight additional charge of \$98.00.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient-- Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Object Module Support-- Access functions and externals in object modules produced by your compiler. **New:** We are now supporting **Microsoft 3.0, Mark Williams & Aztec C** in addition to C.I. C86 & Lattice.
- Complete Multiple Module Support.
- Symbolic Debugging-- Set breakpoints, single-step, and directly execute C expressions.
- Many more features including batch mode and 8087 support.

• **Price: \$300.00 (Demo \$45.00) MC, VISA**

Price of demo includes documentation & shipping within U.S. residents add 6% sales tax. Specify compiler.

- C-terp runs on the IBM PC (or compatible) under DOS 2.x with a suggested minimum of 256Kb of memory. It can use all the memory available.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 (Computer Innovations), Lattice (Lattice Inc.), IBM (IBM Corp.), C-terp (Gimpel Software), Microsoft (Microsoft), Aztec (Manx)

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
158	AMPRO Computers Inc.	125	110	Micro Interfaces Corp.	43
176	Advent Products Inc.	64-65	136	Microcomputer Systems Consultants	42
221	Alcyon Corp.	52	*	Micromint Inc.	84
121	Arity Corporation	117	105	MicroProcessors Unlimited	101
224	Atari Corporation	C-3	*	Microtec Research	113
216	Atron	24	190	Mitek	97
159	Blaise Computing	4	*	Mix Software	59
161	Borland International	C-4	128	Morgan Computing Company	23
126	Boston Software Works	101	79	Mystic Canyon Software	119
181	C User's Group	98	137	OES Systems	127
209	Certified Software Corp.	97	124	Optotech	1
178	Chalcedony	81	192	Overland Data Inc.	50
81	Cogitate Inc.	98	200	Pecan Software Systems	13
122	CompuView	39	76	Personal Tex Inc.	50
225	Computel Publishing Society	116	139	Phoenix Computer Products	46
96	Computer Innovations	72	91	Phoenix Computer Products	5
82	Creative Programming	58	193	Plu Perfect Systems	24
86	Dalsoft Systems	94	169	Poor Person Software	105
130	Data Base Decisions	88	*	Precise Electronics	111
203	Datalight	97	201	Prentice-Hall Inc.	98
87	Digital Research Computers	53	196	Pro/Am Software	54
204	Disclone	83	140	Productivity Products Int'l.	73
179	Earth Computers	86	143	Programmer's Shop	35
89	Ecosoft Inc.	31	141	Programmer's Shop	63
210	Educational Microcomputer Systems	28	205	Quelo	94
90	Edward K. Ream	37	107	Quilt Computing	101
138	Essential Software	77	206	Raima Corp.	19
165	Everest Solutions	C-2	145	Rational Systems Inc.	92
180	Executive Systems	17	170	Relational Database Systems	82
211	FOG	75	213	68 Micros	111
93	Faircom	56	78	SLR Systems	52
94	Fox Software Inc.	22	114	Seidl Computer Engineering	125
*	Gimpel Software	128	217	Semantic Microsystems	33
*	Gimpel Software	48	85	SemiDisk Systems	99
97	Greenleaf Software Inc.	123	212	Sextant	67
98	Guidance Software	107	202	Simon & Schuster Computer Books	42
132	Harvard Softworks	91	219	Simon & Schuster Computer Books	41
134	HiSoft	94	83	Soft Advances	95
*	House Ad (Back Issues)	80	*	Soft Focus	83
*	House Ad (Bound Volume)	120	113	SoftCraft	2
*	House Ad (C Products)	114-115	88	Softaid	109
*	House Ad (Mac Reprint)	113	197	Software Masters	20
*	House Ad (Sourcebook)	122	155	Solution Systems	56
*	House Ad (Subscription)	80	153	Solution Systems	74
*	House Ad (Z80 Toolkit)	100	152	Solution Systems	74
166	Inetco Systems	127	147	Solution Systems	71
194	InfoPro Systems	33	164	Spruce Technologies	121
*	Integral Quality	93	172	Sunny Hill Software	112
215	Kadak Products	36	173	TLM Systems	49
100	Kriya	125	174	TLM Systems	45
101	Lattice Inc.	57	175	TLM Systems	47
135	Lugaru Software, Ltd.	51	150	Trio Systems	37
218	MacMemory Inc.	105	207	Turbo Power Software	85
223	Manx Software	87	214	US Software	113
222	Manx Software	89	120	/usr/group - UniForum	69
109	Manx Software	96	77	UniPress Software	15
108	Manx Software	7	154	Vance Info Systems	36
102	Mark Williams	11	157	Vermont Creative Software	83
189	Martian Technologies	109	112	Wendin Inc.	9
199	Media Cybernetics	55	116	Wizard Systems	43
220	Metadigm Inc.	90	208	Wordtech Systems	21

* This advertiser prefers to be contacted directly: see ad for phone number.

Advertising Sales Offices

East Coast

Walter Andrzejewski (617) 567-8361

Midwest

Michele Beaty (317) 875-8093

Northern California/Northwest

Lisa Boudreau (415) 424-0600

Southern California/AZ/NM

Beth Dudas (714) 643-9439

Advertising Director

Shawn Horst (415) 424-0600



Apple Mac 512™



IBM PCAT™



Commodore Amiga™

THERE'S ONLY ONE WORD FOR THESE PRICES: RIP-OFF.

Introducing the Atari 520ST personal computer system. \$799.95 complete.*

Go ahead. Compare those other machines with the new Atari 520ST™. They cost hundreds of dollars more, but you don't get much in return. That's what we call a rip-off.

For \$799.95* the 520ST comes complete with high-resolution monochrome

monitor, 2-button mouse, 3.5" disk drive, TOS™ Operating System, including GEM™ Desktop, plus Logo™ and Atari BASIC programming languages. \$200 more gives you an RGB color monitor with 512 glowing colors.

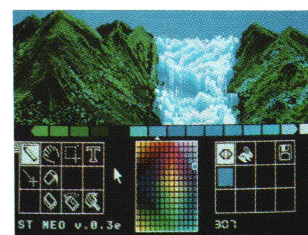
Choose innovative business, entertainment, education, systems management, and integrated package software. Expand your 520ST with industry standard parallel printers, modems, MIDI controlled synthesizers and key-

boards, 1 megabyte floppies, 10 MB and larger hard disks, and more. All available now. At remarkably low prices.

So, go ahead. Compare the ST system to those other guys. Only Atari gives you so much. For so little.

For the dealer nearest you, write Atari Corp., Customer Services, 1196 Borregas Ave., Sunnyvale, CA 94086.

*Plus applicable local taxes. \$999.95 with color monitor.
All prices are manufacturer's suggested retail list.



	ATARI™ 520ST	IBM™ PCAT™	APPLE™ Macintosh™	COMMODORE™ AMIGA™
Price	\$799	\$4675	\$2795	\$1795
CPU	68000	80286	68000	68000
Speed MHz	8.0	6.0	7.83	7.16
Standard RAM	512K	256K	512K	256K
Number of Keys	95	95	59	89
Mouse	Yes	No	Yes	Yes
Screen Resolution (Non-interlaced Mode)				
Color	640 x 200	640 x 200	None	640 x 200***
Monochrome	640 x 400	720 x 350**	512 x 342	640 x 200***
Color Output	Yes	Optional	None	Yes
Number of Colors	512	16	None	4096
Disk Drive	3.5"	5.25"	3.5"	3.5"
Built-in Hard Disk (DMA) Port	Yes	Yes	No	No
MIDI Interface	Yes	No	No	No
No. of Sound Voices	3	1	4	4

**With optional monochrome board (non bit-mapped)
***Interlace Mode - 640 x 400



ATARI®
Power without the price.

IBM & PCAT are registered trademarks of International Business Machines Corp. Commodore & Amiga are trademarks of Commodore Electronics LTD. Apple & Macintosh are trademarks of Apple Computer, Inc. GEM is a trademark of Digital Research, Inc. Atari, TOS & Logo are trademarks of Atari Corp.

SAVE OVER 30% ON OUR GIFT PACKS!
60-DAY MONEY-BACK GUARANTEE

How Borland's Three New Holiday Packs Will Fill Your Stocking Without Emptying Your Piggybank.

Three special packs with dazzling discounts that will help get you into a Holiday mood. You can get some of Turbo, most of Turbo, or all of Turbo—including the two newest members of the Turbo family, Turbo GameWorks™ and Turbo Editor Toolbox™. You also get our unmatched 60-day money-back guarantee, quality products that aren't copy-protected.

TURBO NEW PACK \$95.00.

You get the two exciting new members of the Turbo Pascal family,

- TURBO GAMEWORKS, Chess, Bridge, and Go-Moku, complete with source code and a 200-page manual.
- TURBO EDITOR TOOLBOX, all the building blocks to make your own editors and word processors, complete with source code and a 200-page manual.

TURBO HOLIDAY PACK \$125.00.

You get all three of the Turbo family classics for only \$125.00 (about a 30% discount). Turbo Pascal 3.0 and Turbo Tutor and Turbo DataBase Toolbox—all for just \$125.00.

- TURBO PASCAL combines the fastest Pascal compiler with an integrated development environment.
- TURBO TUTOR teaches you step-by-step how to use Turbo Pascal with commented source code for all program examples on diskette.
- TURBO DATABASE TOOLBOX offers three problem-solving modules for your Turbo Pascal programs: Turbo Access, Turbo Sort, and GINST, which generates a ready-to-run installation program that lets you forget about adapting your software to specific terminals.

TURBO HOLIDAY JUMBO PACK \$245.00.

This is it—the whole thing, the entire Turbo family including its two newest members. You get:

- Turbo Pascal
- Turbo Graphix Toolbox
- Turbo Tutor
- Turbo DataBase Toolbox
- Turbo GameWorks **NEW!**
- Turbo Editor Toolbox

and you pay only \$245.00 for all six! Which means that you're getting everything at only about \$40 a piece. Quite a holiday deal. (And if you already own one or several members of the Turbo family, be creative—nothing can stop you from buying the Jumbo Pack, picking out the ones you already have and giving the rest as holiday gifts to family and friends. At these prices you can afford to give to others and to yourself.) Speaking of Holidays, this offer lasts until March 31, 1986. (At Borland, we like to make the Holidays last.)



4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

Copyright 1985 Borland International BI-1017

Turbo Pascal and Turbo Tutor are registered trademarks and Turbo DataBase Toolbox, Turbo Graphix Toolbox, Turbo Editor Toolbox, Turbo GameWorks, and MicroStar are trademarks of Borland International, Inc. WordStar is a trademark of MicroPro International Corp. Multi-Mate is a trademark of Multi-Mate International Corp. Microsoft is a registered trademark and Word is a trademark of Microsoft Corp. WordPerfect is a trademark of Software Software International.

NEW! TURBO GAMEWORKS \$69.95.

Our new Turbo GameWorks offers games you can play and replay without Turbo Pascal or revise and rewrite with Turbo Pascal 3.0. We give you the source code, the manual, the diskettes and the competitive edge. Chess, Bridge and Go-Moku. State-of-the-art games that let you be player, referee, and rules committee all at once because you have the Turbo Pascal source code. Learn exactly how the games are made—so you can go off and make your own. And Turbo GameWorks is the only quality game you can buy that is not copy-protected. Sold separately, only \$69.95. (Just \$47.50 if you buy the Turbo New Pack.)

NEW! TURBO EDITOR TOOLBOX \$69.95.

Build your own word processor—for only \$69.95!

You get ready-to-compile source code, a full-featured word processor that looks and acts like WordStar™, and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs. With Turbo Editor Toolbox you can have the best of all word processors. You can make WordStar behave like Multi-Mate. Support windows just like Microsoft's Word. And do it as fast as WordPerfect does it. Incorporate your new "hybrids" into your programs to achieve incredible control and power. Sold separately, only \$69.95. (If you buy the Turbo New Pack, the price drops to just \$47.50.)

Holiday Gift Packs, Turbo GameWorks™ & Turbo Editor Toolbox™

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133.

The holiday packs include an upgrade coupon for both options so you get BCD and 8087 support for \$39.95 (regularly \$55.00).

Carefully describe your computer system!

My computer's name and model is: _____
Mine is: ☐ 8-bit ☐ 16-bit
I use: ☐ PC-DOS ☐ MS-DOS
☐ CP/M-80 ☐ CP/M-86

The disk size I use is: ☐ 3 1/2" ☐ 5 1/4" ☐ 8"

Name: _____
Shipping Address: _____
City: _____ Zip: _____
State: _____
Telephone: _____

NOTE: Turbo Editor Toolbox and Turbo GameWorks are available for the IBM PC and true-compatibles using Turbo Pascal 3.0 ONLY.

***Gift Pack
Offers Last
Until March 31,
1986**

	Quantity
* Turbo Holiday Jumbo Pack	\$245.00
* Turbo Holiday Pack	\$125.00
* Turbo New Pack	\$95.00
Pascal	\$69.95
Pascal w/8087	\$109.90
Pascal w/BCD	\$109.90
Pascal w/8087 and BCD	\$124.95
Turbo DataBase	\$54.95
Turbo Graphix	\$54.95
Turbo Tutor	\$34.95
Turbo Editor	\$69.95
Turbo GameWorks	\$69.95

These prices include shipping to all U.S. cities. All foreign orders add \$10 per product ordered.

Amount: (CA add 6% tax) _____
Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft
Credit Card Expiration Date: _____
Card # _____

COD's and Purchase Orders will not be accepted by Borland International. California residents: add 6% sales tax. Outside USA: add \$10 and make payment by bank draft, payable in U.S. dollars drawn on a U.S. bank.